5-2021

Automated conversion of MultiCellDS Digital Cell Lines and ISA-Tab filesets.

Connor Joseph Burns
University of Louisville

AUTOMATED CONVERSION OF
MULTICELLDS DIGITAL CELL LINES AND ISA-TAB FILESETS

By

Connor Joseph Burns
B.S., University of Louisville, 2020

A Thesis
Submitted to the Faculty of the
University of Louisville
J.B. Speed School of Engineering
as Partial Fulfillment of the Requirements
for the Professional Degree

MASTER OF ENGINEERING

Department of Bioengineering

May 2021

AUTOMATED CONVERSION OF
MULTICELLDS DIGITAL CELL LINES AND ISA-TAB FILESETS

Submitted by:
Connor Joseph Burns

A Thesis Approved On

April 26, 2021
(Date)

By the Following Reading and Examination Committee:

Hermann B. Frieboes, Thesis Director

Joseph Chen

Nihat Altiparmak

ACKNOWLEDGEMENTS

Completing this project would not have been possible without the support of many people. Special thanks are due to the following individuals for their impact both on this project and on my life:

To my Thesis Director, Dr. Hermann Frieboes; for encouraging me to pursue a thesis, your guidance in this project, your mentorship throughout my collegiate experience, and your consistent good humor. I was hesitant to take on this project because I had no experience using Python, but I have grown immensely by expanding my knowledge in a new field. This has been a very challenging but rewarding project. Thank you for giving me this opportunity and for your direction along the way.

To Dr. Joseph Chen and Dr. Nihat Altiparmak, for serving on my thesis examination committee.

To Dr. Sam Friedman and Dr. Paul Aiyetan, for fitting me into your busy schedules to discuss the works of this project and for your patience when explaining new concepts. I am very appreciative of your time.

To Lillian Usher and Reed Campana, for your constant support and understanding, along with providing many meals during the final weeks of this project.

Finally, to my parents for always supporting and encouraging me. I would not be where I am today without your efforts to help me grow as a student and a person.

ABSTRACT

Biological simulation tools are a valuable asset to bridge the gap between biological experimentation and biological theory. As systems thinking has become more regularly applied to biological materials and processes, these materials and processes have been characterized in multi-disciplinary, computational manners. This characterization has yielded an ability to create more realistic simulation tools that can provide in silico experiments for a wider range of scenarios. One prominent simulation tool is PhysiCell, an open-source physics-based multicellular simulation program which utilizes input Digital Cell Lines or Digital Snapshots from the MultiCellular Data Standard (MCDS) project.

This project aims to allow data exchange between MCDS Digital Cell Lines and ISA-Tab file sets to bring benefits of online editing and searching held by ISA-Tab files to Digital Cell Lines, which are useful for biological simulation. Additionally, conversion of ISA-Tab file sets to Digital Cell Lines allows a vast increase in the number of cell lines that can be used for simulation within PhysiCell.

This project allows data exchange between the two file types by using Python scripts to convert between the Digital Cell Line (XML) and ISA-Tab (text) file formats. The scripts produced in this project are validated for file format and verified for file contents for the Digital Cell Lines which exist at time of writing (up to DCL #242). All tools and outputs created in this project are open-source and provided on GitHub in order to facilitate cooperation in biological research to increase the knowledge of the scientific community.

In this project, the scope of inputs for ISA-Tab to Digital Cell Line conversion are limited to those which are derivatives of Digital Cell Lines due to limited ability of the conversion script to match ISA-Tab labels to MCDS Digital Cell Line elements. While limited, the created ISA-Tab to MCDS-DCL script provides an accepted ISA-Tab formatting framework for automated conversion rather than requiring file conversion "by hand". The products of this project should be expanded in the future to allow a greater range of ISA-Tab input formats for conversion to Digital Cell Lines.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. Background

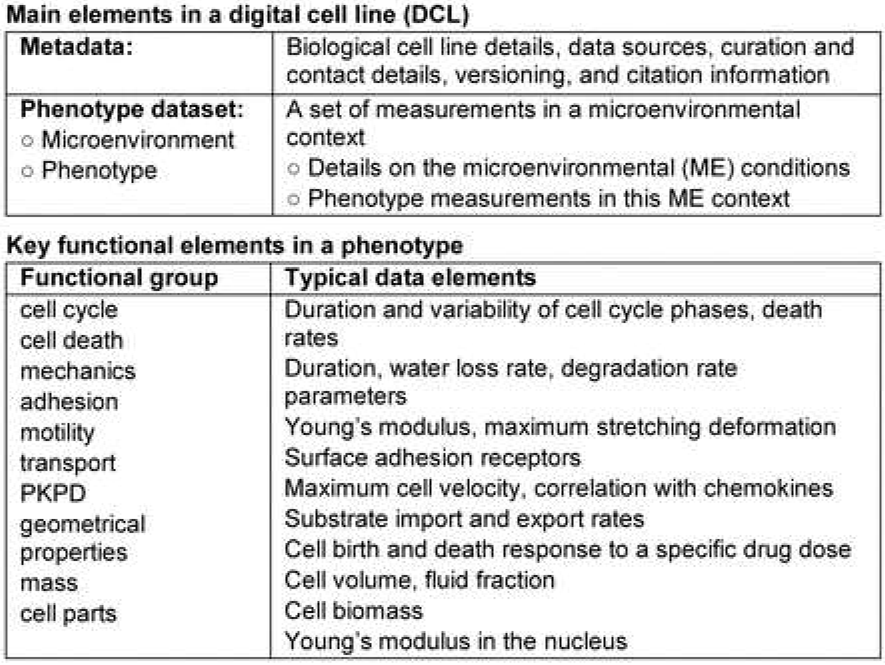
In recent years, computer simulation and modeling have become important tools within the field of biology. Simulations and modeling bring a large benefit to biological research as experiments can be performed “in silico”, bridging the gap between biological theory and traditional biological experimentation (Wellmann). The largest challenge in creating an accurate simulation of biological experiments is that biological processes are extremely complex and vary based on a multitude of conditions (Wolkenhauer and Muir). In order to improve the ability of simulations to accurately reflect outcomes of traditional biological experimentation, a multi-disciplinarian approach called “systems biology” has been adopted on a wider scale. As the name indicates, systems biology focuses on understanding biological processes as a system in which individual interactions are studied in terms of how they affect other processes or materials as a whole. Additionally, the systems biology approach seeks to examine biological processes or materials of interest in a computational manner, through which processes and materials can be described quantitatively (Wake). The systems biology approach lends itself to multi-disciplinarian study because a computational model of a complex system cannot be reduced to traditional biology elements alone: within a cell or organism, electrical, mechanical, physical, and chemical elements are also at play. The movement towards systems biology has yielded biological simulations which aim to capture the complex, multidisciplinary interactions between materials.

One such project that aims to improve biological simulation is PhysiCell, which is described as an "open-source physics-based cell simulator for 3-D multicellular systems" that allows for simulation of "cell fluid and solid volume changes, cycle progression, apoptosis, necrosis, mechanics, and motility" (Ghaffarizadeh et al.). The PhysiCell program allows for user customization of inputs but was created with using data inputs from another project in mind, the MultiCellular Data Standard Project (abbreviated as MultiCellDS or MCDS). The MultiCellDS project is a joint effort between dozens of professionals in the fields of computational biology, biomedical engineering, applied mathematics, oncology, chemical engineering, and systems engineering. Both projects are open-source, which encourages cooperative projects between research groups in order to further increase the knowledge available to the scientific community.

The MultiCellDS Project began to "create a data standard for sharing multicellular experimental, simulation, and clinical data" (Macklin, "The MultiCellular Data Standard Project"). MCDS fills a lack of common format for multicellular simulation data, which otherwise limits the ability to create simulation tools that are shared amongst the scientific community. Additionally, alternative data formats used to record biological experiments and simulations tend to lack machine readability, interoperability with other data formats, or maintenance of biological relationships between data elements (Macklin, "Key Challenges Facing Data-Driven Multicellular Systems Biology"). The MCDS project solves these issues by storing complex biological data in three distinct methods: "digital cell lines, which are analogous to traditional biological cell lines, to record metadata, cellular microenvironment, and cellular phenotype variables of a biological cell line; digital snapshots to consistently record simulation, experimental, and clinical data for multicellular systems; and collections that can logically group digital cell lines and snapshots" (Friedman et al.). MCDS Digital Cell Lines (abbreviated as MCDS-DCL's or DCL's) are the focus of this project.

The MCDS Digital Cell Line file is a method to organize quantitative phenotypical data for one biological cell line in a structured, hierarchical manner. Additionally, DCL's include the microenvironment states in which phenotypical data measurements were taken and metadata including curation information, citation information, and cell line descriptions. Within the DCL framework, phenotypical data elements are separated by the function that is being measured within the cell line. The current standard for DCL's sufficiently represents phenotypical measurement data for "over 200 cell lines, including patient derived and 'standard' cultured cancer cell lines, endothelial cells, yeast, and bacteria" (Friedman et al.). These cell types were chosen to include within the DCL library because each cell type is distinctly different from one another in terms of the phenotypical data and metadata that formulates the cell type's DCL's, allowing for a broad spectrum of biological data to be represented by the MultiCellular Data Standard.

DCL's can be divided into two main parts, metadata of the cell line and phenotype datasets. As previously mentioned, metadata includes data that classify the cell line while phenotype datasets include both phenotypical measurements and the microenvironment conditions in which the measurements were performed (Friedman et al.). For example, the geometrical properties of a glioblastoma cell line may be measured in a low oxygen or hypoxic state. Geometrical property measurements would make up one of many different phenotypical measurement functions within the cell line, and the hypoxic state in which these measurements were taken would be described within the microenvironment conditions. Descriptions that classify the cell line as a glioblastoma cell, conditions of the patient from which the cell originated from, contact information for people involved in creation and curation of the cell line, and a log of curation events would be captured within the DCL's metadata. This structure is demonstrated below in FIGURE 1.



Main elements in a digital cell line (DCL)

Metadata:	Biological cell line details, data sources, curation and contact details, versioning, and citation information
Phenotype dataset: ○ Microenvironment ○ Phenotype	A set of measurements in a microenvironmental context ○ Details on the microenvironmental (ME) conditions ○ Phenotype measurements in this ME context

Key functional elements in a phenotype

Functional group	**Typical data elements**
cell cycle	Duration and variability of cell cycle phases, death rates
cell death	Duration, water loss rate, degradation rate parameters
mechanics	Young's modulus, maximum stretching deformation
adhesion	Surface adhesion receptors
motility	Maximum cell velocity, correlation with chemokines
transport	Substrate import and export rates
PKPD	Cell birth and death response to a specific drug dose
geometrical properties	Cell volume, fluid fraction
mass	Cell biomass
cell parts	Young's modulus in the nucleus

FIGURE 1 – Major DCL Elements (from Friedman et al.)

DCL files are created using the XML (abbreviated for eXtensible Markup Language) file format as XML allows the structuring of data elements in an ordered and hierarchical manner. The hierarchy of major Digital Cell Line elements within DCL XML files is shown below in FIGURE 2. XML provides additional benefit as the DCL file format by being easily readable by humans, having widespread support in other simulation software, and having many related technologies available (Friedman et al.). In order to understand the methodologies used in this project, one must be versed with the basics of XML file structure including elements, nodes, tags, attributes, xPath, and schema. It is recommended that one views XML file structure documentation provided by "W3Schools" if introductions to XML structure or file contents are needed (XML Elements).

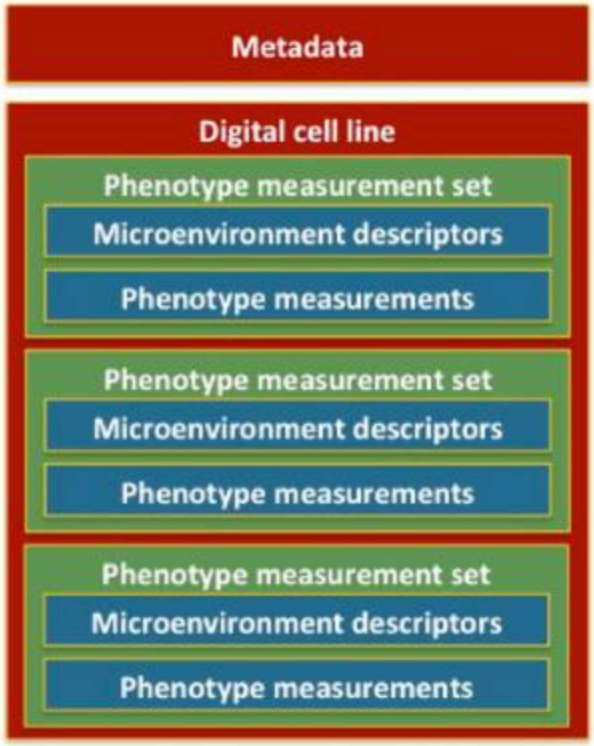

Metadata
Digital cell line
Phenotype measurement set
Microenvironment descriptors
Phenotype measurements
Phenotype measurement set
Microenvironment descriptors
Phenotype measurements
Phenotype measurement set
Microenvironment descriptors
Phenotype measurements

FIGURE 2 – Hierarchy of Major Data Elements of Digital Cell Lines

(from Friedman, "MultiCellDS User Overview")

ISA-Tab (abbreviated for Investigation, Study, Assay, Tab Delimited) is another file format that is used to capture complex biological experiment measurements along with contextual metadata, sample characteristics, and measurement types. ISA-Tab stores data in a set of three file types: The Investigation file, which contains information that describes the overall experiment; the Study file, which contains information relating to the characteristics and treatments applied to the experiment subject; and the Assay file, which defines and provides measurements of data (Rocca-Serra). A concept map of ISA objects and entities, along with the relationships between them, is shown below in FIGURE 3.

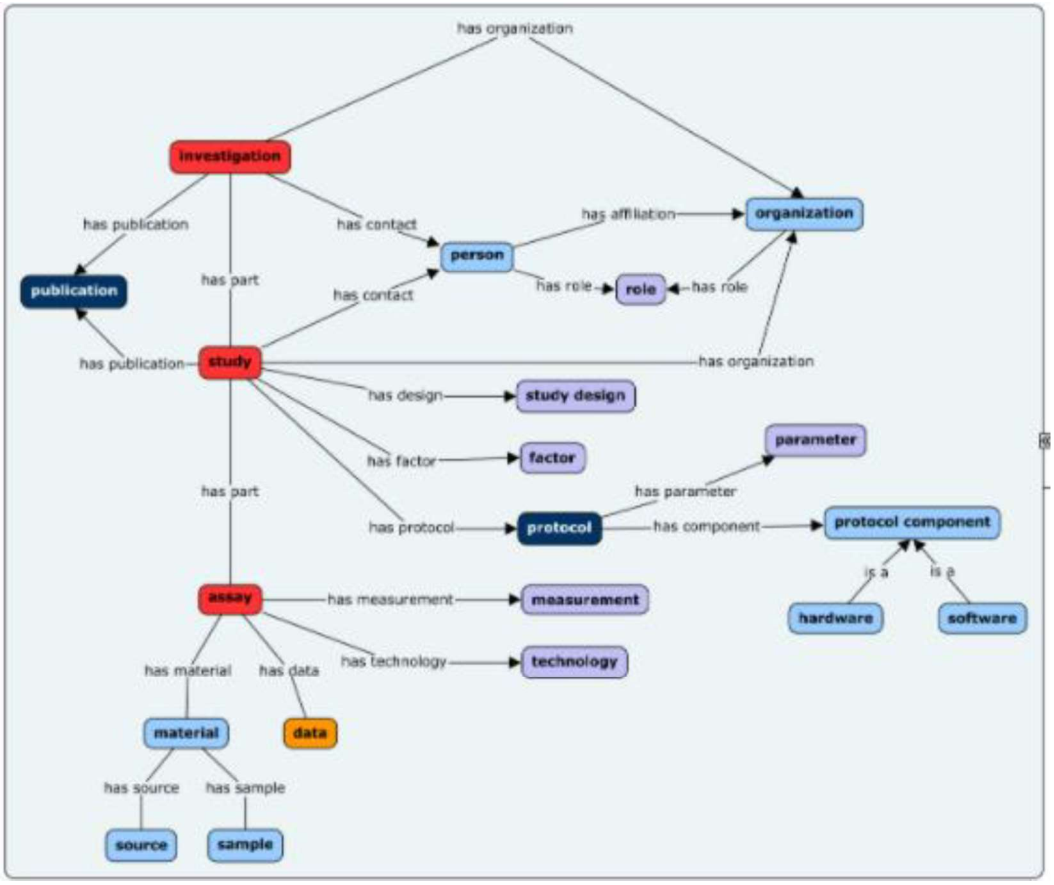


FIGURE 3 – ISA Abstract Model (from “ISA Abstract Model”)

The structure of ISA-Tab files is hierarchical in the sense that data is stored within three files: however, the structure is inherently different from MCDS-DCL files as ISA-Tab uses a text file format. The ISA-Tab text file contains data fields that are separated by rows and by tabs for data fields within the same row. It can be helpful to visualize the data fields that are separated by tabs as different columns. The tab separation in this file type functions similarly to a comma in a comma separated variable (csv) file, which allows conversion of ISA-Tab files into Excel files. When converted to an Excel file, the separation of data fields by tab is translated into columns and the separation of data fields by row is maintained. ISA-Tab is a file standard which has been more widely adopted for storing of biological data. For example, the current MCDS-DCL library is limited to 235 Digital Cell Lines, whereas over 2,000 life/biomedical science datasets in ISA-Tab form are included in the GigaDB data repository ("GigaDB Datasets").

However, ISA-Tab cannot be directly used as a file input for the PhysiCell multicellular simulation program, which is currently the only simulation program of its kind. In order to promote data reusability while increasing the library of MCDS Digital Cell Lines, this project's goal is to create Python scripts that convert data between the MCDS Digital Cell Line and ISA-Tab file types. Completion of this objective allows for a significantly larger amount of multicellular data to be simulated within PhysiCell, increasing the usefulness of the simulation for biological research initiatives. Additionally, creation of conversion scripts enables researchers that document biological research within an ISA-Tab format to create a PhysiCell simulation that is specific to their own research experiment.

B. Existing Work and Project Contents

At the beginning of the project, a partially functioning MCDS-DCL to ISA-Tab conversion script had been created using Python 2. The adjective "partially" is used as the script does not produce a valid ISA-Tab file set from an input MCDS-DCL and many data elements within input DCL files are not translated to the output ISA-Tab file sets. The existing Python 2 script may be referred to as the "old" MCDS-DCL to ISA-Tab conversion script during this paper. The old conversion script utilized relationship mapping within the script to find data elements at a certain xPath within the input DCL, then write the data element to an ISA file under a specified label or header. Data loss in translation by this script is evident by the output Assays not containing all phenotypical measurement elements that exist within the input DCL's (see TABLE VII, "Old Script Outputs").

As previously mentioned, the objective of this project is to create Python scripts for converting from MCDS-DCL to ISA-Tab and from ISA-Tab to MCDS-DCL. Python is used as the coding language for file conversion because it supports libraries for processing both XML and text file formats, is well documented and supported, and is relatively easy to learn (see TABLE V, "Python"). The specification for the MCDS-DCL to ISA-Tab conversion script is that the script produces validated ISA-Tab outputs with no data loss for existing DCL files. The specification for the ISA-Tab to MCDS-DCL conversion script is that the script produces validated MCDS-DCL outputs with no data loss for ISA-Tab files, with input ISA-Tab file sets limited to those which have been created by MCDS-DCL to ISA-Tab conversion. The scope of inputs for this conversion script are limited as ISA-Tab offers a large degree of user customization within Study and Assay files. This makes creating a conversion script that accepts any input ISA-Tab file set a challenge that exceeds this project. However, creating a script that converts ISA-Tab file sets of a specified format (i.e., one that is created by MCDS-DCL to ISA-Tab conversion) still advances the aims of the MCDS project. This script makes creating a new DCL file from ISA-Tab inputs possible as a user can move ISA-Tab data to a format that can be handled by the script. This can be accomplished by reformatting ISA-Tab data and modifying ISA labels. Additionally, successful creation of this script provides a foundation for creating a script that accepts more input forms in the future.

Completing these objectives requires an expansion and modification of data element relationship mapping between the two file types. These relationships are stored in an Excel file since this format is easy to read. The relationships Excel file is also designed to hold additional information that dictate conversion script operations. This approach is a marked improvement over the process used in the old script, as the Excel approach allows for the new conversion scripts to handle a wider range of data elements while simplifying the process of updating the conversion scripts when DCL's containing new data elements are added to the DCL library.

II. PROCEDURE

For the completion of this project, a particular order of methods, analysis, and testing is followed. An overview of the process is shown below in FIGURE 4. Each step within the process overview flow chart will be discussed in greater detail during the following sections. Python scripts and Excel files that were created to accomplish the goals of this project are publicly available in a repository hosted by GitHub (see TABLE VII). Additionally, a text file (see TABLE VII, “requirements.txt”) containing the external Python libraries used in this project (see TABLE VI) in an installable format was generated using pigar (see TABLE VI , “pigar”). This streamlines library set-up for future users of the conversion scripts. For development and testing of Python scripts, PyCharm was used as an integrated development environment (IDE) (see TABLE V, “PyCharm (Community Edition) ”). The ElementTree module from the lxml library is used to translate XML file data into memory within Python scripts (see TABLE VI, “lxml”).

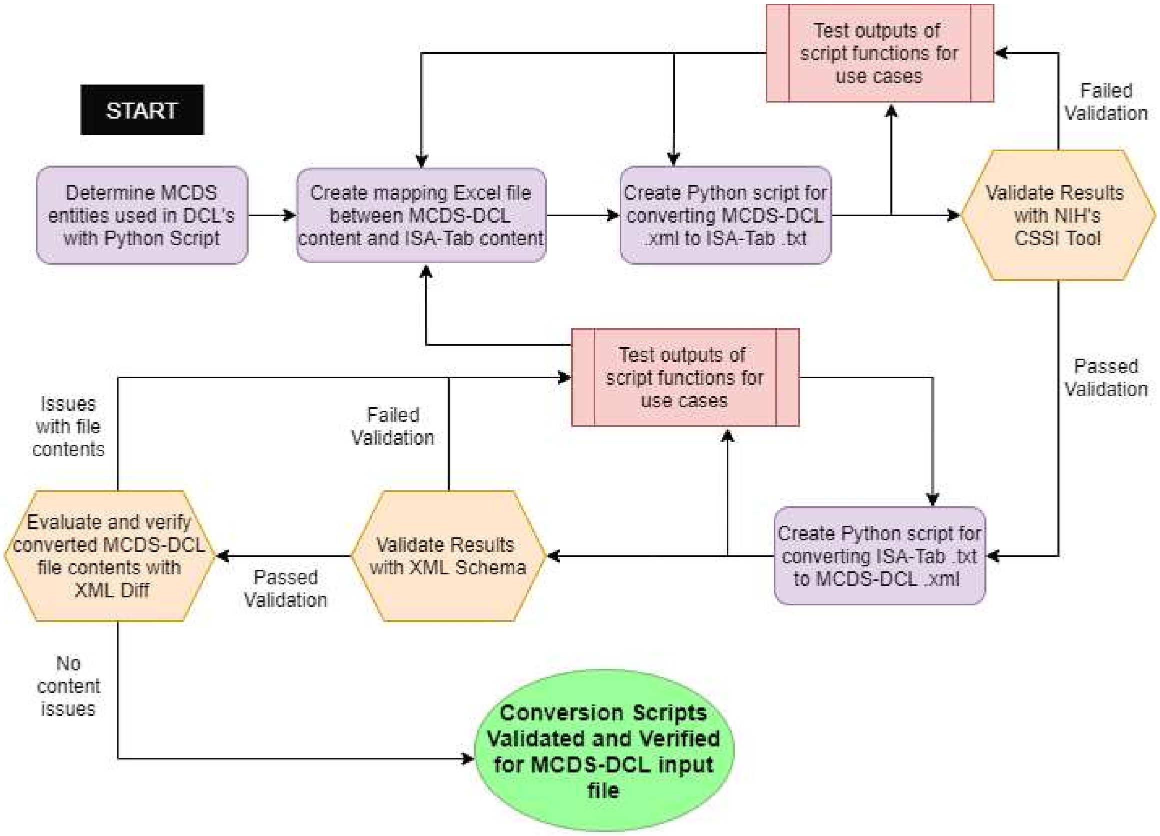

START
Determine MCDS entities used in DCL's with Python Script
Create mapping Excel file between MCDS-DCL content and ISA-Tab content
Create Python script for converting MCDS-DCL .xml to ISA-Tab .txt
Test outputs of script functions for use cases
Failed Validation
Validate Results with NIH's CSSI Tool
Passed Validation
Test outputs of script functions for use cases
Issues with file contents
Failed Validation
Evaluate and verify converted MCDS-DCL file contents with XML Diff
Passed Validation
Validate Results with XML Schema
Create Python script for converting ISA-Tab .txt to MCDS-DCL .xml
No content issues
Conversion Scripts Validated and Verified for MCDS-DCL input file

FIGURE 4 – Project Overview: Procedural Flowchart

A. Determining MCDS-DCL Content and Mapping Relationships

The first step in constructing conversion scripts is determining what data is used within MCDS Digital Cell Lines. The MultiCellular Data Standard supports 1,440 different XML data elements and a total of 7,215 attributes under XML elements, resulting in a total of 8,655 distinct locations in which data can be stored. Many of the attribute names repeat for different XML elements, but nonetheless this would be a very large amount of data to map. For the remainder of this section, attributes of the same name that appear under different element tags are discussed as different attributes. In order to reduce the amount of data mapping necessary to complete project objectives, a Python script is used to parse the 235 existing DCL's and create an Excel file that contains the elements and attributes of all DCL's (see TABLE VII, "mcds_content_list.py"). This script operates by creating a pandas DataFrame that contains each XML element, each attribute under the element, the file in which the element is found, the xPath of the element, an indication of the data held by the element, and an indication of whether the element or attribute is used multiple times within the input DCL (see TABLE VI , "pandas"). After parsing the DCL files, operations are performed on the DataFrame to analyze the data. These operations include creating two additional DataFrames: one in which duplicates of elements and attributes within the same file are removed, and one in which duplicates of elements and attributes within the collection of DCL's are removed. Additionally, elements which do not hold any text or attribute data are removed from these two DataFrames. The resulting third DataFrame functions as a list of all discrete elements that hold text and/or attribute data within the collection of DCL's. If an element ever appeared as a multiple within the DCL file set, it is labeled as a multiple in this DataFrame. Each of these DataFrames is then written to an Excel file to save the results (see TABLE VII, "MCDS_DCL_All_Content.xlsx").

Next, a copy of this Excel file is made to prevent overwriting and used to perform further analysis. The analysis performed includes calculating the number of times an element tag appears in the file set, as well as the number of files that the element tag appears in. These metrics are created for distinct xPaths as well. The result quantitatively describes the appearance of tags and elements within the file set (see TABLE VII, "MCDS_DCL_DataLog.xlsx"). Additionally, this Excel file is used to document the usage of the DCL content within conversion scripts as a means to ensure that all content is accounted for and provide a relative location of where content conversion occurs within a conversion script.

Through determining MCDS-DCL entities, it was discovered that five DCL's required updating. Three updates were due to the use of a custom tag in place of an element tag established within MCDS. One of these updates required subsequent updating of the MultiCellDS schema, as the element tag was documented but not yet included in the schema because it is only used in one DCL (see TABLE VII, "MultiCellDS.xsd"). The remaining two updates were due to files missing an attribute that declares the XML file as a DCL. For the remainder of this project, the updated versions of these five files were used in place of the original files. The original files, file names of updated versions, and changes made to create updated versions are documented in an Excel file (see TABLE VII, "DCL_xml Updates.xlsx").

A final step for MCDS-DCL content processing is determining the set of DCL's that encapsulate all discrete elements and attributes of the file set. This is accomplished by removing duplicate file names from the Excel sheet containing discrete DCL elements and attributes. Since the duplicate removal processes in both Python and Excel keep the first instance of an entity, the remaining files are the first instance of DCL content within the file set with respect to numerical order. The remaining files comprise a set of seven DCL's, which are displayed below in TABLE I. The resulting list of discrete locations that existing DCL's store data contains a total of 322 elements and attributes. The reduction from over 8,000 locations to 322 locations allows a much more efficient mapping process to be used, while ensuring that all existing DCL content is accounted for. The table containing all data locations within DCL's is not included due to size, but a subsection of the table is provided as an example in TABLE VIII.

TABLE I

DCL SET CONTAINING CONTENT LOCATIONS OF ALL DCL'S

MCDS-DCL File Names
MCDS_L_0000000003.xml
MCDS_L_0000000044.xml
MCDS_L_0000000049.xml
MCDS_L_0000000066.xml
MCDS_L_0000000238.xml
MCDS_L_0000000240.xml
MCDS_L_0000000241.xml

Now that the MCDS-DCL data has been processed, mapping relationships are created between elements and attributes of the MCDS-DCLs and ISA-Tab format. Relationship mapping is accomplished through application of the ISA-Tab specification guidelines to the elements and attributes of DCL's. Although ISA-Tab stores data in text files, there is a degree of data hierarchy maintained through the division of data into Investigation, Study, and Assay files. Additionally, data hierarchy can be maintained through ISA labels and organization of data by columns or rows.

The process of translating MCDS-DCL data to an ISA-Tab format involves dividing the MCDS-DCL data into sections by a particular node, determining which element tags may repeat within the node, and determining what the node describes or measures. This determination process is accomplished through a mixture of inspecting example DCL files in Notepad++, utilizing xPaths of elements and attributes in the MCDS_DCL_DataLog.xlsx file, and basing decisions on MultiCellDS documentation (Friedman, “MultiCellDS Version 1.0.0 Full Documentation”), (Friedman, “MultiCellDS User Overview”). In general, data within the metadata node of MCDS translates to the Investigation file, as the Investigation file is used “to record metadata” in addition to linking “related Study objects under an Investigation” (“ISA-Tab Format”). Data within the phenotype dataset nodes of MCDS-DCL’s describe experimental conditions and measurements taken during experiments. Therefore, the child nodes of the phenotype dataset are translated to Assays, which represent a “test performed…producing qualitative or quantitative measurements” (“ISA-Tab Format”). The objective of the Study file is to relate the Assay files to one another by “contextualizing information for one or more Assay” and to “record the provenance of biological samples” (“ISA-Tab Format”). Therefore, MCDS elements that describe the collection of phenotype datasets or characteristics of the cell line such as the cell origin element translate to the Study file.

However, there are many challenges associated with determining relationships between MCDS and ISA-Tab. One particular challenge in determining relationships is structuring many-to-one relationships. Consider the example of presenting multiple people’s contact information: in MCDS, this is accomplished by using a “role” node that describes a person’s role in the creation of the MCDS-DCL and child nodes that contain the person’s contact information in element text. In ISA-Tab, contact information is spread out among separate rows, with quotations surrounding each person involved in the Study. In this instance, data of the same hierarchical level in the MCDS-DCL file (contact information) needs to be changed to an array format in order to fit the ISA framework.

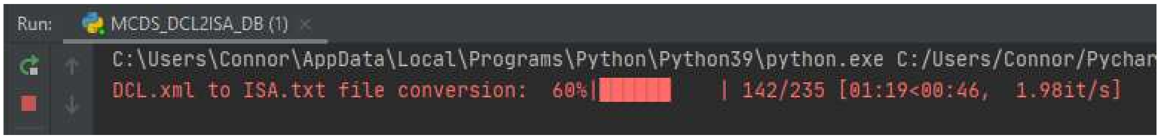
Furthermore, there are differences in the data contained by each file. In a MCDS-DCL, the person’s role is expressed as an element tag with the remaining data contained as text values under elements; the person’s first and middle initials are joined together into a single text value under the element “given-names”; and both the organization-name and department-name fields refer to a person’s affiliation. Each of these examples differs from the way that ISA-Tab stores contact data.

For many data fields, establishing relationship mapping is a mixture of creative art and engineering. There is not a hard and fast way to relate data fields between MCDS-DCL’s and ISA-Tab: what is important is that the file conversion creates validated output files with minimal data loss, and that relationships are created in a way that is logical both to human and machine reading.

MCDS-DCL and ISA-Tab relationships are defined in an Excel file along with additional information that dictates the decisions made by the conversion script (see TABLE VII, “ISA_MCDS_DCL_Relationships.xlsx”). The benefit of this approach is that updating the data fields which are eligible for conversion may be accomplished without needing to change the script itself; this is a valuable design feature since MCDS is meant to evolve over time and updating a row within an Excel file is a more straightforward, less time-consuming process than updating lines or even large sections of code. Additionally, the relationships need to be documented in a way that is easy to read by people. By defining relationships in an Excel file, both of these desired characteristics are fulfilled.

B. Converting MCDS-DCL XML File to ISA-Tab Text Files

The MCDS-DCL to ISA-Tab conversion script operates by processing information contained in the relationship Excel file to determine where an ISA-Tab entry should be written (which ISA file and the location in that file), what MCDS-DCL data the ISA-Tab entry correlates to, the location of the correlate DCL data within the input XML file, and how the data should be treated. The pandas library is used to import data from the relationship Excel file into the Python script for usage. The tqdm module of the tqdm library (see TABLE VI, "tqdm") is used to provide a status bar when running the conversion script that indicates how many files have been converted out of the total number of input files, estimated total conversion time, and estimated time remaining (see FIGURE 5).



```
C:\Users\Connor\AppData\Local\Programs\Python\Python39\python.exe C:/Users/Connor/Pychar
DCL.xml to ISA.txt file conversion:  60%|██████    | 142/235 [01:19<00:46,  1.98it/s]
```

FIGURE 5 – Tqdm Status Bar

1. Overall Script Structure

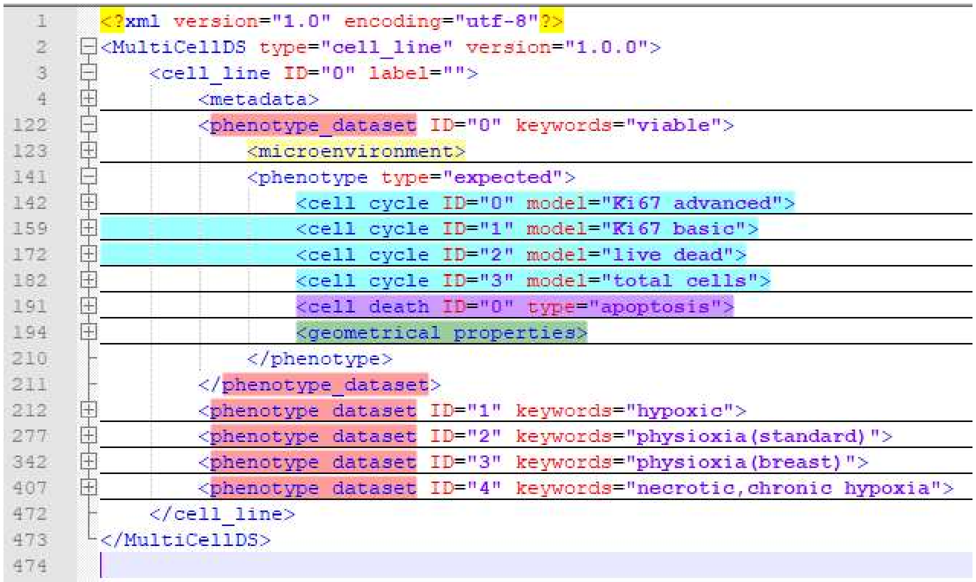
The MCDS-DCL to ISA-Tab conversion script can be run from an IDE such as PyCharm or from a Python console. The user may specify input DCL's for conversion. Accepted forms are an absolute path to a directory containing conversion files, an absolute path of a DCL file to be converted, or the file name of a DCL within the "All Digital Cell Lines" directory (TABLE VII, "All_Digital_Cell_Lines"). By default, each collection of

Investigation, Study, and Assay files will be written to a folder containing the base name of the MCDS-DCL file, which is the MCDS-DCL file name with the extension removed. The user can specify a directory in which to write the folder containing converted ISA-Tab file sets. The user can also choose to check the input file(s) for unsupported MCDS-DCL elements and attributes. This is accomplished by parsing the input file, determining the xPath for each element with text or attribute data within the input file, and comparing the xPath against a list of supported MCDS-DCL data fields. This list is imported to the script from the relationships Excel file. If the input XML file contains unsupported data fields, the user is notified with a print statement containing a list of the element and attribute locations which are not supported. This feature is useful as it notifies a user if any data may be lost in conversion without needing to verify results. As more Digital Cell Lines are created, using this check will be beneficial for determining whether an update to the relationships Excel file or potentially the script itself is necessary without requiring full verification testing.

Additionally, the script checks that the input file is an MCDS-DCL XML file. If the input file fails this condition, the conversion script skips attempted conversion for the file and notifies the user that an improper file has been used as an input. This keeps an error from occurring if the user provides a directory that contains any files besides DCL's and instead allows conversion to proceed for only the DCL's. During the code development process, checks were performed regularly to ensure that the script functioned properly for different use cases. These regular tests allow coding errors to be discovered before attempting to validate and verify the conversion for all files. For example, after a script function was written to concatenate the MCDS-DCL data fields "organization-name" and "department-name" to form the Investigation file data field "Study Person Affiliation", scenarios in which either or both of the MCDS-DCL data fields were missing were tested to make sure that the desired conversion output for the "Study Person Affiliation" data field was achieved. The operations within the script can be divided into three sections, one for each of the output file types it produces.

2. Converting to Assay Files

The Assay section is a combination of 10 different functions within the script, one for each potential Assay output file. Each Assay is based off of a MCDS-DCL node that contain a quantitative characterization of the cell line. These nodes may be referred to as "Assay nodes". Each node that correlates to an Assay exists under the phenotype dataset node, except for the transitions node (which is a sibling to phenotype dataset and metadata nodes) and the clinical node (which resides under the metadata node). Majority of elements used as the basis for Assays are characterized by an ability to repeat in the input file, since the phenotype dataset node has an ability to repeat. Additionally, some Assay nodes can repeat within the same phenotype dataset node. For example, one phenotype dataset node may house multiple cell cycle nodes. This does not mean that the same elements will exist under nodes of the same tag, either within the same phenotype dataset or in different phenotype datasets. An example of the potentially repetitious structure described is shown in FIGURE 6, which is a screenshot of a MCDS-DCL file with certain nodes collapsed. The file contains 5 phenotype dataset nodes, the first of which is expanded. Each element type underneath the phenotype dataset node that is highlighted in the same color correlates to an Assay file which will be created during conversion. Conversion of this file would create a Microenvironment Assay, Cell Cycle Assay (which contains measurements from each of the cell cycle nodes), Cell Death Assay, Geometrical Properties Assay, and potentially other Assays depending on the contents of the other phenotype dataset nodes. Assay files contain contents from each occurrence of the "Assay node" within the input DCL, regardless of their parent node.



```
<?xml version="1.0" encoding="utf-8"?>
<MultiCellDS type="cell_line" version="1.0.0">
    <cell_line ID="0" label="">
        <metadata>
        <phenotype_dataset ID="0" keywords="viable">
            <microenvironment>
            <phenotype type="expected">
                <cell cycle ID="0" model="Ki67 advanced">
                <cell cycle ID="1" model="Ki67 basic">
                <cell cycle ID="2" model="live dead">
                <cell cycle ID="3" model="total cells">
                <cell death ID="0" type="apoptosis">
                <geometrical properties>
            </phenotype>
        </phenotype_dataset>
        <phenotype dataset ID="1" keywords="hypoxic">
        <phenotype dataset ID="2" keywords="physioxia(standard)">
        <phenotype dataset ID="3" keywords="physioxia(breast)">
        <phenotype dataset ID="4" keywords="necrotic,chronic hypoxia">
    </cell_line>
</MultiCellDS>
```

FIGURE 6 – Phenotype Dataset Division

The separate Assay writing functions all follow the same key steps, represented in the form of a flowchart in FIGURE 7.

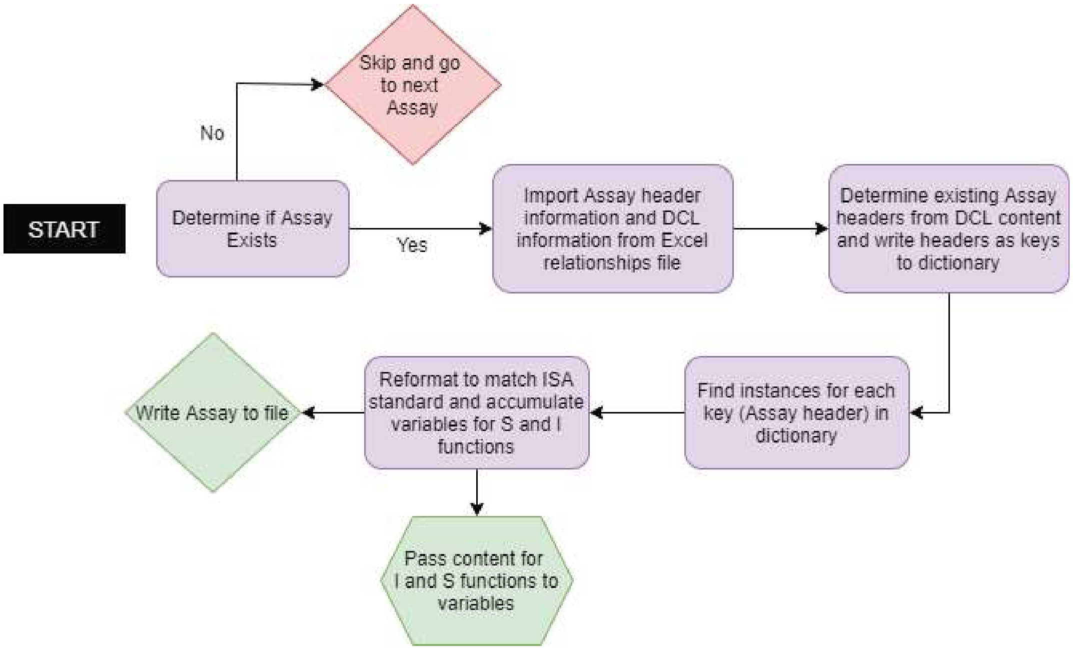

Skip and go to next Assay
No
START
Determine if Assay Exists
Yes
Import Assay header information and DCL information from Excel relationships file
Determine existing Assay headers from DCL content and write headers as keys to dictionary
Write Assay to file
Reformat to match ISA standard and accumulate variables for S and I functions
Find instances for each key (Assay header) in dictionary
Pass content for I and S functions to variables

FIGURE 7 – Flow Chart of Assay File Conversion

First, the determining element for the Assay, which is also the name of the Assay, is searched for within the input DCL by xPath. If there are no occurrences of the element within the input file, the function for the Assay is not executed and the script moves on to the next Assay. If there are any occurrences of the element within the input file, the function for that Assay executes and an Assay file for the element will be generated. Each Assay node element that is found is passed as an argument to the function. There is a worksheet for each Assay within the relationship Excel file containing the relationships between Assay headers and MCDS-DCL element data. This worksheet supplies information to create majority of the Assay header names. Each Assay header name is paired with an associated partial xPath. This partial xPath corresponds to the xPath of elements to search for within the located Assay nodes, with the xPath of the located Assay node removed. This leaves an xPath “tail” that can be appended to the xPath of each Assay node in the file to create combinations of xPaths using a loop in the script. These xPath combinations encapsulate the locations of every possible data element within the Assay nodes. This is a large benefit since Assay nodes may have repeated occurrences within the DCL file. There is also a small amount of information written to each Assay that is located outside of the Assay node. This information includes the “Sample Name”, which is created from concatenating the MCDS cell line name with the ID of the phenotype dataset where the Assay node is located.

Each data row in the produced Assay text file corresponds to an independent Assay node within the DCL. Structuring Assay files this way allows for expression of relationships between Assay nodes within the DCL in addition to fitting the prescribed Assay file form, which specifies that each row should correspond to a distinct sample. Existence of each potential Assay header is tested by determining if data held by the Assay header exists for any row (i.e., any Assay node). If no Assay nodes contain data for the Assay header, the header is not written to the Assay file. This keeps data from being created in conversion, as there is a difference between a measurement not existing at all and the measurement existing within the Assay file but not having any reported values. The second case would indicate that the file may have been translated with data loss or that a researcher has forgotten to include the data.

After determining the existing Assay headers within the file, the script searches for data correlating to each Assay header using a loop. This correlates to the script searching for each combination of one partial xPath with the xPaths of each Assay node. After searching each combination (i.e., determining all rows for the Assay column), the script progresses to the next Assay header and repeats until completion. After determining all data to be converted for the Assay, certain data such as the variables within parameter value headers (indicating a Study Protocol Parameter in the Assay) is saved to a list to be used in writing the Investigation file. The data is processed with pandas to orient the data in proper form for Assay specifications and finally written to the Assay text file.

3. Converting to a Study File

The Study file is created mostly from information held underneath the cell origin node, which provides information describing the subject that the cell line is derived from. This information includes the species, organ, characteristic disease, and BTO ID of the cell line. Additionally, information specific to the patient the cell line is derived from may be included in the Study file. This is common for DCL's of cancer cells. If patient specific information exists, the information is held underneath the custom/clinical node. Custom/clinical information written to the Study file includes the patient's age at diagnosis, the pathology grade of the cell sample, and patient survival metrics. As in converting to an Assay file, a worksheet within the Excel relationships file provides the script with Study file headers and the associated DCL xPath at which to search for data. The elements within DCL's that correlate to Study file headers only exist in a single place within the DCL file: this makes conversion for Study files simple, as the conversion script only needs to search one location for data. The Study conversion function works by searching for data at each provided xPath from the Excel relationships file and writing the data under the specified Study header if the data exists. As in Assay files, the Study file is formatted with headers along the first row of the text file and with data entries along subsequent rows aligned by tab with the appropriate label. After searching for all Study headers, the data is written to the Study text file.

4. Converting to an Investigation File

The DCL data fields that comprise the Investigation file originate from the metadata node of the DCL. Additionally, information from the created Assay and Study files is used to describe each of these files within an Investigation file. In the Investigation file, ISA-Tab labels are aligned along the left side of the text file and written with one label per row, which lends to the conversion script fully processing one row at a time, then moving down a row and beginning conversion for a new ISA-Tab label. This can be thought of as a transpose of the Study and Assay file structure. Again, the functions for Investigation file writing utilize information within the relationships Excel file. Data within the relationships Excel file for the Investigation section is structured similarly to the Investigation file, with each row of the Excel file corresponding to a row that will be written to the Investigation file. Because the Investigation file must contain a number of entities that are specified within ISA-Tab guidelines, each of these rows is written to the output Investigation file, regardless of the row containing converted data fields or not.

There are several decision paths that the script makes based on the information provided in the relationships Excel file. First, a decision is made based on the content of the cell within the MCDS-DCL Correlate X-Path column and the same row as the ISA-Tab label being processed (see FIGURE 8). This cell can hold the xPath(s) at which the script should search for element attribute or text data; "Text Entry", signifying that the text to write in the row is provided in the adjacent column of the Excel file; or "Script Variable", signifying that the text to write is provided in the script. The xPath format is most common for the rows of the Investigation file. The "Text Entry" format is used only for text that is constant among all cell lines, such as the Ontology Source Reference section. The "Script Variable" approach is used for writing information that has been collected from writing the Assay and Study files to rows, such as the Study Assay File Name row. If there is no content within this cell, quotation marks will be written to the row that signify no data is in the row. If the associated "Multiples for xPath" column has a value equal to "Single", one set of quotation marks will be written. If the associated "Multiples for xPath" column has a value equal to "Multiple – (Label Name)", a number of quotation marks equivalent to the number of data fields in the (Label Name) row of the Investigation file will be written. This allows for relationships within sections of the Investigation file to be maintained.

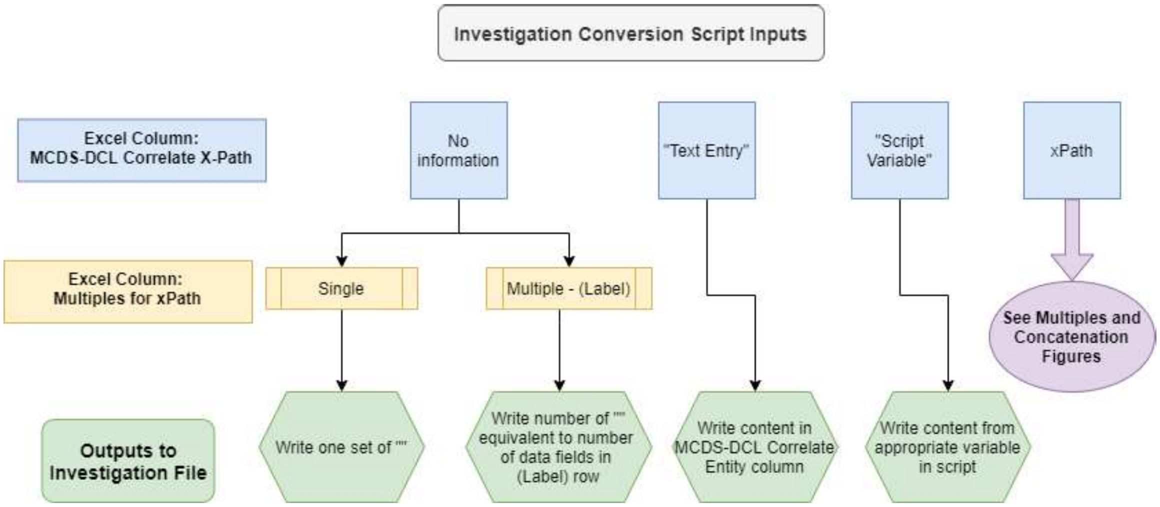

Investigation Conversion Script Inputs
Excel Column: MCDS-DCL Correlate X-Path
No information
"Text Entry"
"Script Variable"
xPath
Excel Column: Multiples for xPath
Single
Multiple - (Label)
See Multiples and Concatenation Figures
Outputs to Investigation File
Write one set of ""
Write number of "" equivalent to number of data fields in (Label) row
Write content in MCDS-DCL Correlate Entity column
Write content from appropriate variable in script

FIGURE 8 – Investigation File Conversion, High Level Decision Tree

There are further decisions made for the xPath input based on the relationships held between correlate MCDS-DCL and ISA-Tab data. Either a single xPath or a list of xPaths separated by comma can be used as an input within this cell. This gives either a single xPath to search for data, or a list of xPaths to search for data. Within this framework, xPaths can be concatenated by a specified variable, indicated with the format xPath1” ConcatenationVariable “ xPath2. For example, an entry xPath1” – “xPath2 would result in data found at xPath1 and xPath2 being written as “Data1 – Data2” within the Investigation file. This approach can be applied to the list of xPaths format as well, in which xPath1” – “xPath2 , xPath3” ! “xPath4 results in data found at the xPaths being written as “Data1 – Data2”   “Data3 ! Data4”. If data is not found at either xPath to be concatenated, then the concatenation variable does not appear, and data found at the single xPath is written in quotation marks (see FIGURE 9).

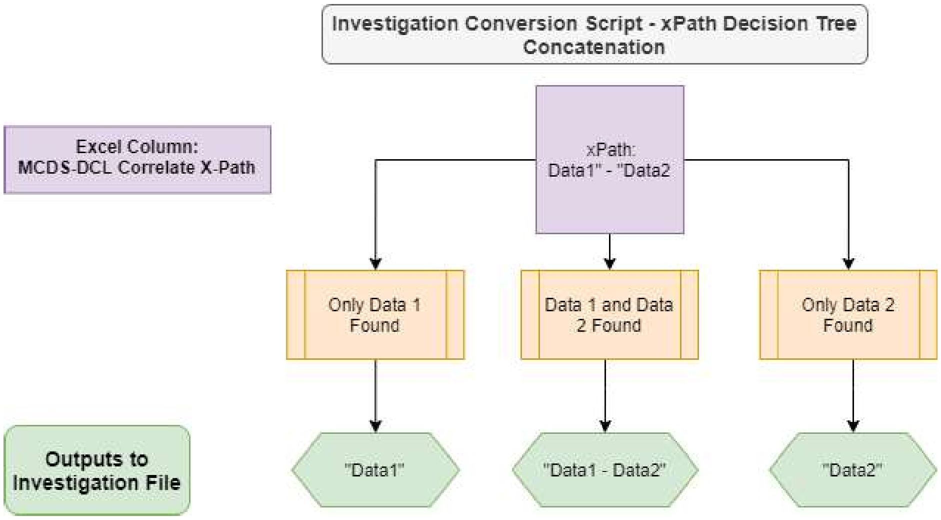

Investigation Conversion Script - xPath Decision Tree Concatenation
Excel Column: MCDS-DCL Correlate X-Path
xPath: Data1" - "Data2
Only Data 1 Found
Data 1 and Data 2 Found
Only Data 2 Found
Outputs to Investigation File
"Data1"
"Data1 - Data2"
"Data2"

FIGURE 9 – Investigation Conversion, xPath Decision Tree for Concatenation

Additionally, a decision is made to search for all data elements that share the same xPath or to only retrieve data from the first element at the specified xPath. This decision is made based on the Excel sheet entry “Multiple”, “Mult-Join”, or “Single”. If the entry is “Single”, the script will only accept data from the first instance of the xPath. If the entry is “Multiple”, the script will accept data from all instances of the xPath within the DCL and each data entry will be written as a separate ISA data entity. If the entry is “Mult-Join”, data from all instances of the xPath will be joined with a semicolon as one ISA data entity (see FIGURE 10).

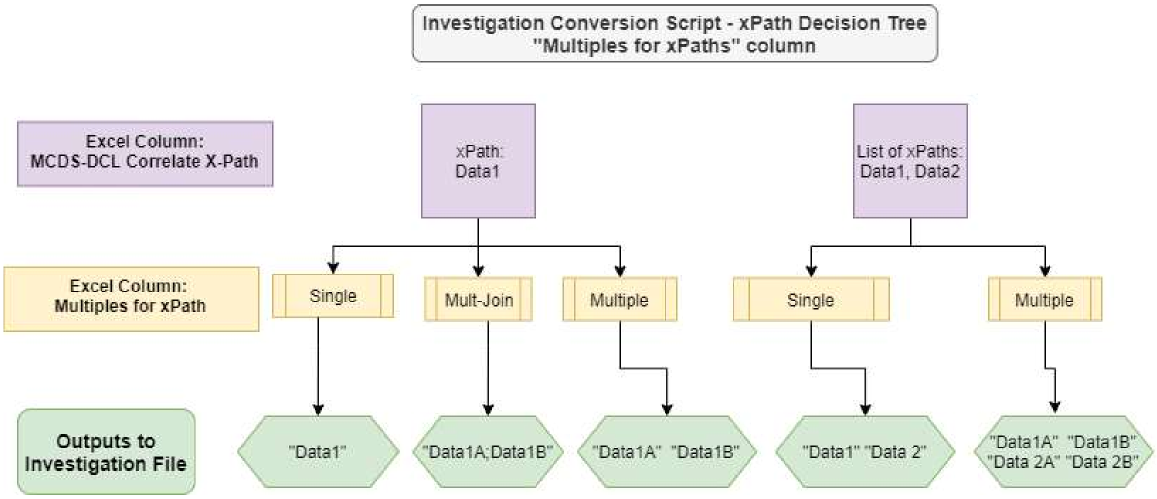
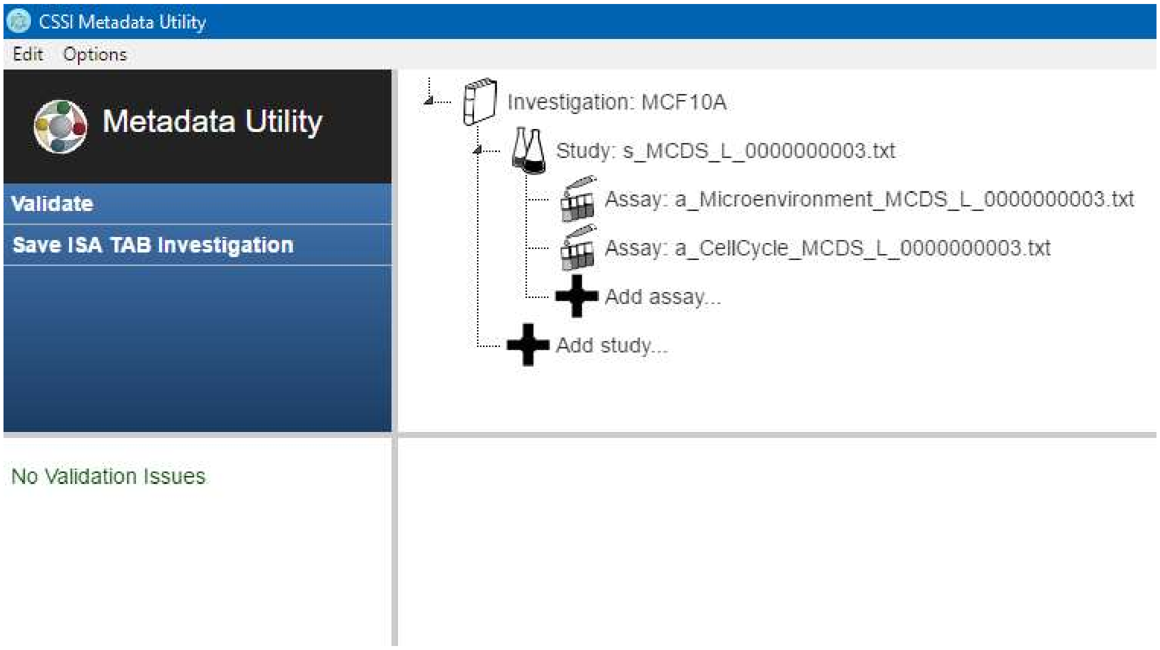


FIGURE 10 – Investigation Conversion, xPath Decision Tree for Multiples

After passing through all rows of the Investigation file, “Comment” rows that have no data entry are removed from the Investigation file. This is done to reduce clutter, since the “Comment” rows are custom ISA-Tab labels created to contain DCL data that does not have an appropriate location within the specified Investigation file framework. Any row specified within the Investigation file framework that does not have any data entries is left in the file with the appropriate number of quotation marks in order to meet ISA-Tab guidelines.

C. Validating Created ISA-Tab Files

Validation of created ISA-Tab files is accomplished using the “Validate” function of the Center for Strategic Scientific Initiatives (CSSI) Metadata Utility software, developed by the National Cancer Institute of the National Institute of Health, or NIH/NCI (“Validating Metadata…”). This validation tool ensures that the imported file is compliant with ISA-Tab standards and displays a window indicating whether validation issues are detected or not. If validation issues are detected, a description of the errors is also displayed. This software is utilized by importing the created Investigation output file for a MCDS-DCL. The CSSI software then searches in the same directory as the Investigation file for the Study and Assay file names indicated in the Investigation file and imports the additional files. Next, the “Validate” function is selected to validate the files and display results of the validation test. An example of a validation test with no errors is shown below in FIGURE 11, and an example of the results window for a failed validation test with errors included is shown below in FIGURE 12.



Investigation: MCF10A

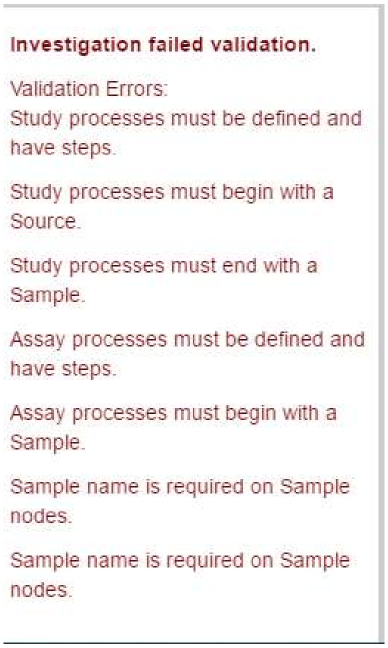
Study: s_MCDS_L_0000000003.txt

Assay: a_Microenvironment_MCDS_L_0000000003.txt

Assay: a_CellCycle_MCDS_L_0000000003.txt

No Validation Issues

FIGURE 11 – CSSI Validation Tool, Passed Validation Window



Investigation failed validation.

Validation Errors:
Study processes must be defined and have steps.

Study processes must begin with a Source.

Study processes must end with a Sample.

Assay processes must be defined and have steps.

Assay processes must begin with a Sample.

Sample name is required on Sample nodes.

Sample name is required on Sample nodes.

FIGURE 12 – CSSI Validation Tool, Failed Validation Window

Because validation testing is a manual process in terms of loading files and reading results, and validation can only be performed on one ISA-Tab file set at a time, the collection of 235 ISA-Tab file sets converted from input MCDS-DCL's would take a considerable amount of time to validate. Therefore, validation testing is performed on a selected collection of ISA-Tab output file sets. This collection is comprised of converted ISA-Tab file sets from DCL's that encapsulate all elements containing text and attributes in existing DCL's (see TABLE I). By using this test set, the length of validation testing time is decreased from many hours to minutes while a high degree of confidence is maintained that the conversion script output file sets will validate for all other existing Digital Cell Lines. Since the validation test results are displayed in the CSSI software window but not output from the program, results of validation for each selected ISA-Tab output file set are recorded in an Excel table.

D. Converting ISA-Tab Text File Sets to a MCDS-DCL XML File

Since the scope of ISA-Tab to MCDS-DCL file conversion is limited in this project to the ISA-Tab files generated from MCDS-DCL's, the ISA-Tab to MCDS-DCL conversion script operates in a similar manner to the MCDS-DCL to ISA-Tab conversion script, but in the reverse direction. In this script, information contained in the relationship Excel file is processed to determine where data stored in the ISA-Tab file set should be written within the MCDS-DCL file and how that data should be treated. This relationship is used by finding the desired ISA-Tab label/header within the input Investigation, Study, or Assay file and writing the data it contains to the associated xPath. The pandas library is used to import data within the relationship Excel file into the Python script for usage. The tqdm module from the tqdm library is used to provide a status bar when running the conversion script that indicates how many file sets have been converted out of the total number of input file sets, estimated total conversion time, and estimated time remaining (see FIGURE 5). As in the MCDS-DCL to ISA-Tab conversion script, this script can be run from an IDE or Python console. Users can specify an absolute path of the desired input ISA-Tab file set as well as a directory location for writing the output DCL XML file(s).

1. Creating a MCDS-DCL Template

A MCDS-DCL template file is supplied to the script for writing ISA-Tab file contents to the output MCDS-DCL. The MCDS-DCL template file is an XML file containing all discrete elements and attributes in the existing DCL file set with element text and attribute values removed. Since all ISA-Tab files to be converted in this project are derived from DCL's, all elements and attributes to which converted ISA-Tab data can be written are encapsulated by this template file. The template file is generated using a Python script to remove all text and attribute values from copies of the set of DCL files that hold all elements and attributes of the existing DCL's (see TABLE I). Contents from the seven "cleared" XML files are then pieced together in one XML file until all DCL elements and attributes are contained in the file. This process is checked by using the MCDS-DCL template file as an input to the mcds_content_list.py script to generate an Excel document of xPaths for all elements and attributes within the template file, then comparing the resulting list of xPaths against the list of all discrete xPaths in existing DCL's (see TABLE VII, "MCDS_DCL_DataLog.xlsx"). This comparison is completed using the

"COUNTIFS" function in Excel to determine if any xPaths in the list of all discrete xPaths in existing DCL's are missing from the list of xPaths in the template XML. Once the template XML contains all DCL xPaths, the ordering of element tags and attributes is altered to match the ordering prescribed by MCDS schema documentation (Friedman, "MultiCellDS Version 1.0.0 Full Documentation"). This process involves manually copying and pasting elements within the DCL template XML to match the ordering of elements within the schema documentation. Validation of the DCL template file with the MCDS schema cannot be completed since the template file has no text or attribute values, which would cause the file validation to fail; however, if the template file contents are sequenced in a way that violates the schema, no converted MCDS-DCL output file would pass validation. If all converted MCDS-DCL output files failed validation, there would be an indication of a potential issue with the DCL template XML.

2. Conversion from ISA-Tab to MCDS-DCL

The script progression used to convert an input ISA-Tab file set to a MCDS-DCL file is expressed as a flowchart below in FIGURE 13.

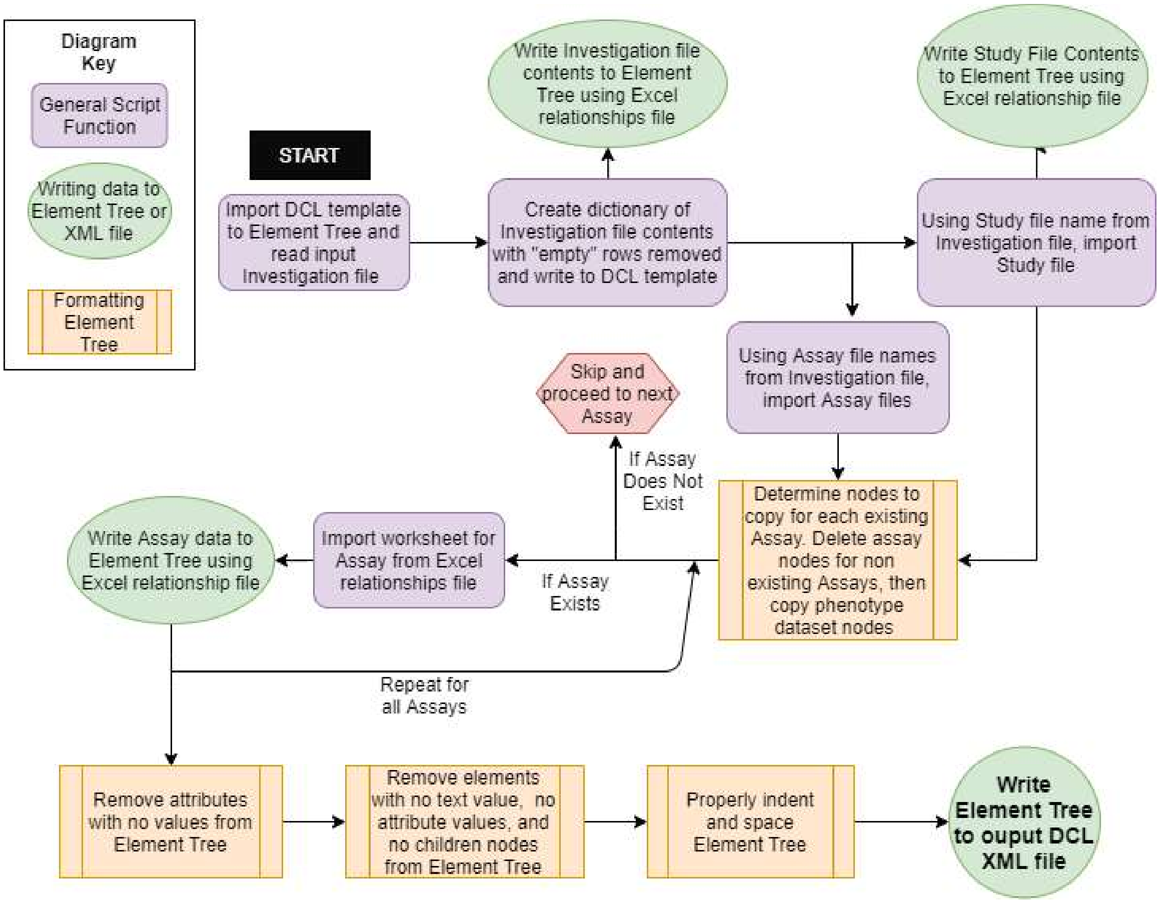


Diagram Key
General Script Function
Writing data to Element Tree or XML file
Formatting Element Tree
START
Import DCL template to Element Tree and read input Investigation file
Write Investigation file contents to Element Tree using Excel relationships file
Create dictionary of Investigation file contents with "empty" rows removed and write to DCL template
Write Study File Contents to Element Tree using Excel relationship file
Using Study file name from Investigation file, import Study file
Using Assay file names from Investigation file, import Assay files
Skip and proceed to next Assay
If Assay Does Not Exist
Determine nodes to copy for each existing Assay. Delete assay nodes for non existing Assays, then copy phenotype dataset nodes
Write Assay data to Element Tree using Excel relationship file
Import worksheet for Assay from Excel relationships file
If Assay Exists
Repeat for all Assays
Remove attributes with no values from Element Tree
Remove elements with no text value, no attribute values, and no children nodes from Element Tree
Properly indent and space Element Tree
Write Element Tree to ouput DCL XML file

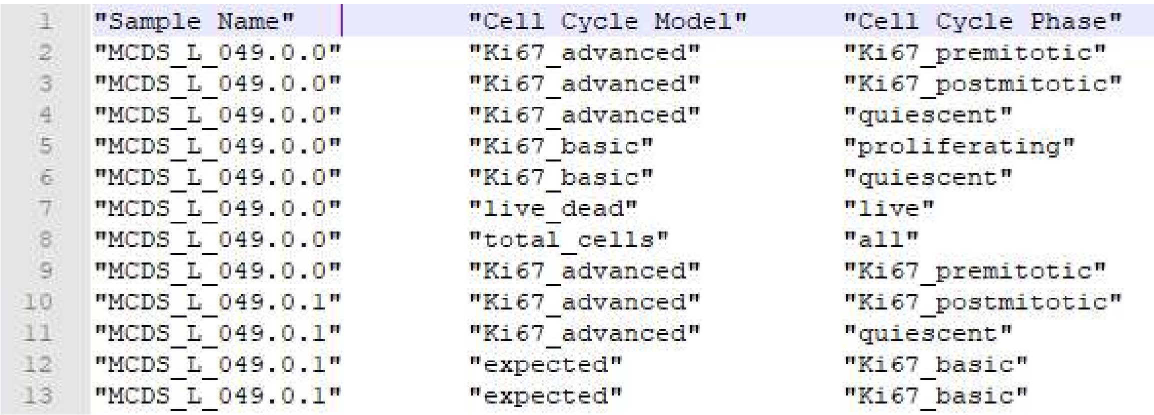
FIGURE 13 – ISA-Tab to MCDS-DCL Conversion Script Progression

To begin, the DCL template XML and the Investigation file from the input ISA-Tab file set are imported into the program. The contents of the template XML are saved to an lxml Element Tree, while the rows of the Investigation file which have content in addition to the row label are saved to a dictionary. Using the relationships Excel file, Investigation file rows which only have one data field are written to the Element Tree by matching the row label to the appropriate xPath in the relationships Excel file. Investigation file rows with multiple data fields are processed separately since they are placed at different nodes. Rows with multiple data fields are processed by the Investigation subsection from which they originate. For example, each row of the "Study Contacts" section is processed together.

This is done because “columns” of text data within the sections match to one another – each “column” within the section is created from the same parent DCL node, so it is written back to the DCL in this manner. During multiple row processing, there is also a determination made of whether multiple nodes of the same tag exist. If there are multiple nodes with the same tag, copies of the original node are made and placed within the Element Tree. This is completed before writing data to these Element Tree nodes so that data written to the node is not copied. Unintentional data copying could result in values existing where there should be no value since the field would not be overwritten in the Element Tree, resulting in incorrect data being added to the converted DCL.

Rows from the Investigation file provide the names of the Study and Assay files within the file set. These rows are used to import the Study and Assay files as pandas DataFrames from the file set directory. Since ISA-Tab is a tab-delimited format, ISA files can be read as csv files with a tab as the delimiter. After importing the Study file, contents within the Study DataFrame are written to the Element Tree. This is accomplished by looping through the columns of the Study DataFrame, finding the Study header within the Study worksheet of the Excel relationships file, finding the xPath associated with the Study header, and writing the value contained in the column of the Study DataFrame (imported Study file) to the associated xPath (from the relationships Excel file).

A similar process is used for each Assay file; however, all Assay files are imported before writing any data to the Element Tree. This is done so that the nodes of the Element Tree are properly formatted for the ISA-Tab data which will be written to the nodes. If any of the 10 Assay files do not exist, the associated Assay node is removed from the Element Tree (ex. Microenvironment element for the Microenvironment Assay file) in order to reduce working memory. For Assay files that do exist, all Assay nodes are determined for the Assay file and copied if necessary. For example, the Cell Cycle Assay may contain multiple cell cycle Assay nodes within a phenotype dataset, and multiple cell phase Assay nodes within a single cell cycle node. This is shown below in FIGURE 14, which is a Cell Cycle Assay file in which names and structure are modified for viewing. The "Sample Name" header corresponds to a phenotype dataset within the file, while the "Cell Cycle Model" header corresponds to a cell cycle node and the "Cell Cycle Phase" header corresponds to a cell phase node. In the DCL file, there must be three cell cycle phase nodes since there are three different cell cycle phases (Ki67_premitotic, Ki67_postmitotic, quiescent) under a single cell cycle model (Ki_67 advanced). There are also four distinct cell cycle nodes indicated by "Cell Cycle Model" (Ki67_advanced, Ki_67 basic, live_dead, total_cells) that belong to one phenotype dataset (MCDS_L_049.0.0, or phenotype dataset ID#0). Therefore, a total of three cell phase nodes will be created in the Element Tree by copying the existing cell phase node. Next, a total of four cell cycle nodes will be created in the Element Tree by copying the existing cell cycle node. The copying process is completed in order from child element to parent element for Assays with multiple Assay nodes so that the maximum number of child nodes that can exist within one parent node will exist in each parent node. Copying nodes in this manner ensures that nodes that are necessary for writing ISA-Tab data to the XML file will exist in the Element Tree. Each Assay node is collected in a dictionary to use for writing Assay files to the Element Tree.

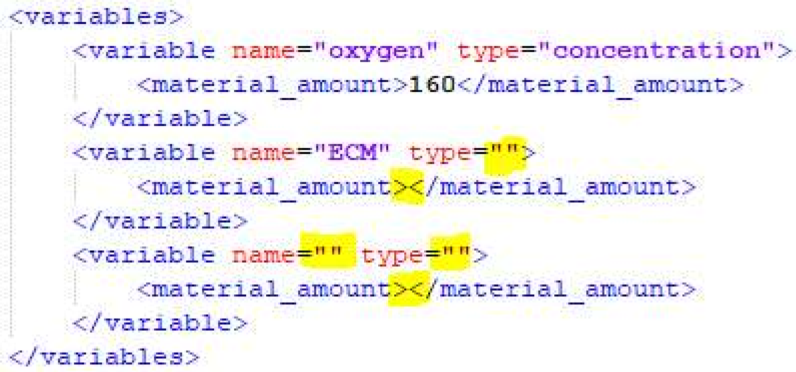


```
"Sample Name"           "Cell Cycle Model"      "Cell Cycle Phase"
"MCDS_L_049.0.0"        "Ki67_advanced"         "Ki67_premitotic"
"MCDS_L_049.0.0"        "Ki67_advanced"         "Ki67_postmitotic"
"MCDS_L_049.0.0"        "Ki67_advanced"         "quiescent"
"MCDS_L_049.0.0"        "Ki67_basic"            "proliferating"
"MCDS_L_049.0.0"        "Ki67_basic"            "quiescent"
"MCDS_L_049.0.0"        "live_dead"             "live"
"MCDS_L_049.0.0"        "total_cells"           "all"
"MCDS_L_049.0.0"        "Ki67_advanced"         "Ki67_premitotic"
"MCDS_L_049.0.1"        "Ki67_advanced"         "Ki67_postmitotic"
"MCDS_L_049.0.1"        "Ki67_advanced"         "quiescent"
"MCDS_L_049.0.1"        "expected"              "Ki67_basic"
"MCDS_L_049.0.1"        "expected"              "Ki67_basic"
```

FIGURE 14 – Modified Cell Cycle Assay Nodes

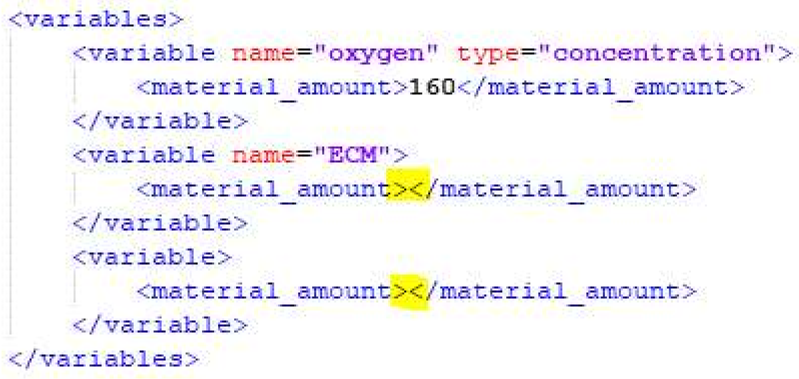
After “initializing” the Element Tree with all necessary nodes, data from each Assay is written to the Element Tree using a similar approach to converting the Study file, in which the Assay header is found from the relationship Excel file, the associated xPath is extracted, and the data within the Assay file is written to the extracted xPath. The main difference in converting Assay data is that data may exist at different nodes and/or at nodes with the same tag that are repeated within the Element Tree. Therefore, an extra step occurs in which xPaths of Assay nodes collected in the dictionary are matched to appropriate rows of data within the Assay file. This step methodically places data at the same node from which it originated before converting to ISA-Tab form. The Assay conversion process repeats for each Assay file within the file set. After all Assays have been converted, all data has been written to the Element Tree.

After writing all data to the Element Tree, the Element Tree is “cleaned” in preparation for writing to the output XML file. This process is completed because the template XML file from which the Element Tree is created contains every element and attribute that may occur in a DCL. Therefore, each Element Tree will contain empty attributes or tags after converting ISA-Tab data unless every element and attribute has data transferred to it. An example of how an Element Tree may have attributes with no content is shown in FIGURE 15, with empty attributes and empty element text fields highlighted. The "cleaning" process consists of first removing all attributes in the Element Tree which hold no value. An example of the change that occurs within the Element Tree after removing empty attributes is shown in FIGURE 16. Next, elements are removed from the Element Tree if they meet all of the following criteria: having no text content, having no attribute content, and having no child nodes. This effectively removes all elements that have no data besides the element tag itself from the Element Tree. An example of the change that occurs in the Element Tree after removing empty elements is shown in FIGURE 17.



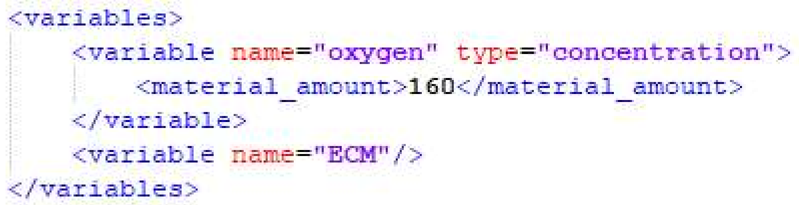
```
<variables>
    <variable name="oxygen" type="concentration">
        <material_amount>160</material_amount>
    </variable>
    <variable name="ECM" type="">
        <material_amount></material_amount>
    </variable>
    <variable name="" type="">
        <material_amount></material_amount>
    </variable>
</variables>
```

FIGURE 15 – Uncleaned XML Section



```
<variables>
    <variable name="oxygen" type="concentration">
        <material_amount>160</material_amount>
    </variable>
    <variable name="ECM">
        <material_amount></material_amount>
    </variable>
    <variable>
        <material_amount></material_amount>
    </variable>
</variables>
```

FIGURE 16 – XML Section, Empty Attributes Removed



```
<variables>
    <variable name="oxygen" type="concentration">
        <material_amount>160</material_amount>
    </variable>
    <variable name="ECM"/>
</variables>
```

FIGURE 17 - XML Section, Empty Elements Removed

At this point, the Element Tree is properly cleaned since attributes holding no data are removed and all elements that hold no data are removed. The final step before writing the Element Tree to the output XML file is properly tabbing nodes of the Element Tree. This is accomplished by looping through each node of the element tree, counting the number of parent levels that a node falls under, and placing that number of tabs before the node. This orients each element of the Element Tree in proper location when writing to XML, which improves human readability. Finally, the Element Tree is written to an XML file with a name equivalent to the original DCL base name with "_converted.xml" added to signify that the file has been converted to ISA-Tab and back to a DCL XML again.

E. Validating and Verifying Converted MCDS-DCL Files

Validation and verification of converted MCDS-DCL files is accomplished using a script created in Python (see TABLE VII, "convertedDCL_eval.py"). This script takes a converted MCDS-DCL file or folder containing converted MCDS-DCL files as an input, along with the absolute path of the folder which contains the original MCDS-DCL files (All Digital Cell Lines folder). Additionally, a hyperlink to the MCDS schema file set on GitLab is provided in an error message if the MCDS schema files cannot be located. This message allows users to easily download the MCDS schema files for DCL validation if necessary. As with the conversion scripts, tqdm is utilized to provide users a status bar.

The script first validates the input converted DCL file by using the XMLSchema module from the xmlschema library (see TABLE VI, "xmlschema"). This module uses the MCDS schema files to check that the DCL's contents follow the specified structure of the schema. If there are any validation issues, the module returns "False". If the DCL passes validation with no issues, the module returns "True". Next, the original DCL is located within the folder of MCDS-DCL files by finding a match to the converted DCL's base name (the converted DCL file name with "_converted" removed). The original DCL is parsed into an Element Tree and empty elements are removed from the Element Tree. Empty elements are removed so that differences between the files are not identified due to the nonexistence of empty elements in the converted DCL. This is desirable because empty elements hold no data. The converted DCL is parsed into a separate Element Tree so that the two files' contents can be compared. Before comparison, the converted DCL is passed into a function which counts the number of nodes and attributes within the file. This number correlates to the total number of places within the file that hold data values. Next, the Element Tree of the converted DCL is compared against the Element Tree of the original DCL. This is accomplished using the diff_trees module from the xmldiff library (see TABLE VI, "xmldiff"). This module compares each element within the element trees and produces errors if there is a difference between the location, attributes, or content of the element. The error output contains an action that would be taken to match the two Element Trees (ex. "update-attribute" if the attributes did not match), the xPath where the difference occurs, and the value of the difference.

Certain differences are removed from the error output, as they signify a difference between the documents that is not a content issue. There are two cases of differences that are removed from the error output, the first of which is the original DCL containing whitespace after an element. This means that certain elements within the original Element Tree have spacing that does not follow the traditional XML format, in which a new element appears on the following line and is either outdented by one tab if it is a parent to the previous element, indented by one tab if it is a child to the previous element, or equivalently tabbed if it is a sibling to the previous element. Because this is a difference due to unstandardized formatting in the original file and not a difference in data content, it is removed from the set of file differences. The second difference removed from the set of file differences is the original DCL containing "nan" (not a number) as an attribute or element text value where the converted DCL does not. This issue is removed because it is not a true loss in data content in addition to being unavoidable during conversion. By default, pandas places "nan" in any DataFrame cell that is empty. This results in an inability to determine whether there is a value of "nan" in a DataFrame created in file conversion because “nan” is a placeholder for an empty cell or “nan” is in the DCL as a text or attribute value. Therefore, “nan” values are not written to converted files. This is not problematic as “nan” is not a useful data value with regards to DCL elements to begin with. “nan” does not clearly represent whether a measurement is taken and the output is not expressible in a number form (example: Infinity), whether the measurement is taken but missing (example: N/A) or whether no measurement was taken (example: None). Each of the example placeholders would be a better value to use in place of “nan” due to decreased ambiguity. Since this difference is expected to occur in a select number of DCL elements, the difference is removed from the set of errors.

The testing process detailed above is repeated for each converted DCL within the input folder if a folder is provided instead of a single file. After completing testing for all input converted DCL’s, output results from the testing process are written to an Excel file (see TABLE VII, “DCLConversionEval.xlsx”). The first worksheet contains the name of each converted file, the results of validation against the MCDS schema, the total number of elements and attributes holding data within the DCL, and the number of differences between the converted DCL and the original DCL. The second worksheet contains the xmldiff module output for any differences in the two DCL’s along with the filename of the converted DCL in which the difference occurred. This conversion evaluation Excel file can be analyzed to validate the ISA-Tab to MCDS-DCL conversion script and verify both conversion scripts. While only converted DCL files are verified for contents, the matched ISA-Tab file set is verified if the converted DCL file content is verified. This deduction is made because all data fields in converted files are found within the original DCL, except for those which are constant for all DCL files (ex. File type attribute value of “cell_line”) or for all generated ISA-Tab files (Ontology References). If there is any data loss when converting a DCL to ISA-Tab the data loss would also be apparent in the converted DCL, triggering a difference to be reported in the evaluation Excel file. Moreover, verification of content within a converted ISA-Tab file set cannot be performed because there is no standard for the ISA-Tab file to be compared against. There is no way to compare the converted ISA-Tab file to the original DCL file without converting it to an XML file.

III. RESULTS AND DISCUSSION OF RESULTS

A. Validation Testing

Validation of converted ISA-Tab files was performed on the collection of file sets as described in the PROCEDURE section, with results shown below in TABLE II. All file sets passed validation with the CSSI Metadata Utility software, which supports validation of the MCDS-DCL to ISA-Tab conversion script for existing DCL's since the collection of file sets encompasses all elements and attributes that hold data values within the set of existing DCL's. Additionally, ISA-Tab file sets were converted from the same input DCL files using the old conversion script. Each tested ISA-Tab file set failed verification testing, which demonstrates the improvements in file conversion gained through creating the new conversion script. A numerical measure of validation errors for ISA-Tab file sets created with the old conversion script was not recorded because errors only show up once in the error display window, regardless of how many times they occur within the input ISA-Tab file set. Therefore, providing this measure would be misleading as one error occurring 12 times in the file set would be represented the same as one error occurring a single time within the file set.

TABLE II

ISA-TAB OUTPUT CONVERSION FILE VALIDATION

MCDS-DCL File Names	Validation Results for Output ISA-Tab File Set	
	New Conversion Script	Old Conversion Script
MCDS_L_0000000003.xml	Pass	Fail
MCDS_L_0000000044.xml	Pass	Fail
MCDS_L_0000000049.xml	Pass	Fail
MCDS_L_0000000066.xml	Pass	Fail
MCDS_L_0000000238.xml	Pass	Fail
MCDS_L_0000000240.xml	Pass	Fail
MCDS_L_0000000241.xml	Pass	Fail

Validation of the converted DCL files against the MCDS schema was performed as described in the PROCEDURE section for all 235 converted DCL's. Every output for validation testing is "True", indicating that every converted DCL passes validation against the MCDS schema. Therefore, the ISA-Tab to MCDS-DCL conversion script is validated for all existing DCL files. These results are provided in TABLE IX.

B. Verification of File Contents

Verification of converted DCL file contents was performed by comparing converted file contents to file contents of the original DCL as described in the PROCEDURE section for all 235 converted DCL's. The converted DCL files with differences reported from verification testing are shown below in TABLE III. All converted DCL files that are not included in TABLE III had no differences from the original DCL. Full results are provided in TABLE IX.

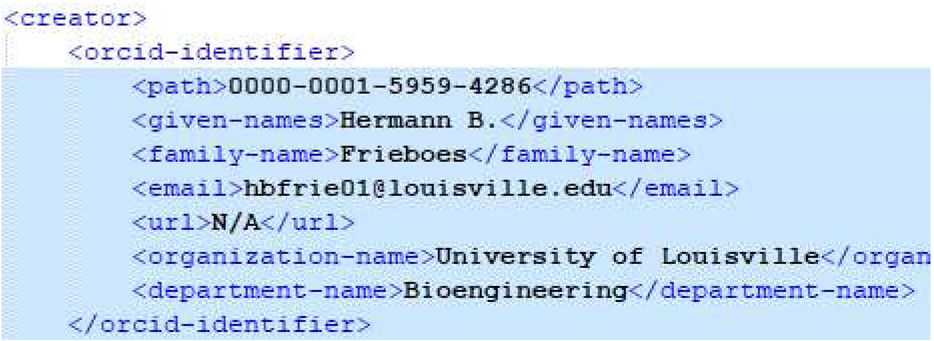
TABLE III

DIFFERENCES BETWEEN ORIGINAL AND CONVERTED DCL'S

Testing Index	DCL Filename	Passed Validation	Total Data Entities in Converted File	Num of Issues in Converted File
39	MCDS_L_0000000044	TRUE	162	2
40	MCDS_L_0000000047	TRUE	162	2
41	MCDS_L_0000000048	TRUE	162	2
233	MCDS_L_0000000240	TRUE	536	98
234	MCDS_L_0000000241	TRUE	102	5
235	MCDS_L_0000000242	TRUE	97	5

Each converted DCL with reported differences from the associated original DCL was analyzed to determine if the difference is an error in file contents using the Excel worksheet containing xmldiff outputs (see TABLE VII, "DCLConversionEval.xlsx") and observing the reported difference within Notepad++ using the Compare plugin.

The first difference reported in DCL file #044 is due to different ordering of contacts under the creator node. In the converted file, the orcid-identifier data for contact "Hermann B. Frieboes" is listed first under the creator node as a result of this person also being the current contact for the DCL file. In the original file, this person's orcid-identifier node appears after two other orcid-identifiers under the creator node. The Notepad++ Compare plugin highlights lines which are equivalent but in a different location in blue. As shown below in FIGURE 18, this difference is highlighted only in blue, which indicates that there is no difference in file content, only ordering. Since the converted DCL passed validation testing, this difference in ordering is acceptable and the difference can be discounted as an error.



```
<creator>
    <orcid-identifier>
        <path>0000-0001-5959-4286</path>
        <given-names>Hermann B.</given-names>
        <family-name>Frieboes</family-name>
        <email>hbfrie01@louisville.edu</email>
        <url>N/A</url>
        <organization-name>University of Louisville</organ
        <department-name>Bioengineering</department-name>
    </orcid-identifier>
```

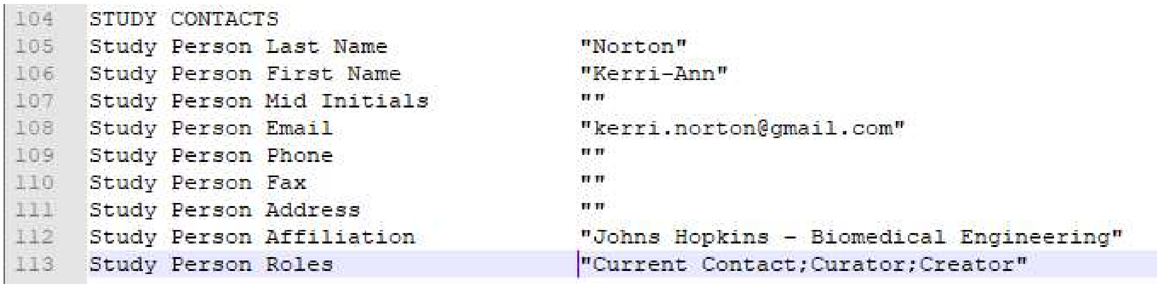
FIGURE 18 – Differences in Contact Ordering

The second difference reported in DCL file #044 is due to the ID attribute being created for the Cell Death node and assigned a number value of 0 in the converted DCL when it does not appear in the original DCL file. As with many other nodes, the ID number of the Cell Death node signifies the order of the specific element within the DCL file if the element tag repeats within the same phenotype dataset. In the original DCL, the ID attribute is not used since there is only one instance of the Cell Death node. It is a good practice to use the ID attribute whenever applicable in case a DCL has additional information appended in the future; therefore, this difference is not considered as an error. The same two differences occurred in #047 and #048 as well. Therefore, each of these three files are fully verified for file contents.

The differences occurring within DCL #241 were matters of spacing and attribute location. There were three differences in this file due to spacing. These differences occurred in long sections of text which are separated into multiple lines (ex. Text under the citation/text location). In each of these three instances, the difference between the converted and original DCL is that the converted DCL uses tabs to align the text block under the appropriate element whereas the original DCL uses spaces. This triggered a difference since the tabbing/spacing is included in the middle of the text with the whole block of text being considered as one text value during evaluation. When compared in Notepad++, no difference was shown between the two files in these areas, indicating that the data content in these areas is equivalent and the spacing differences should not be considered errors. The second type of difference that occurred in DCL #241 is a difference in attribute location for a microenvironment variable. In the converted DCL, the units attribute is located under the variable element with the name attribute “oxygen”. In the original DCL, the units attribute is located under the material amount element, which is a child of the variable element with the name attribute “oxygen”. Both “oxygen” variable elements are located within the same phenotype dataset parent node in their respective files, and both unit attributes hold “mmHg” as the attribute value. In this case, both attributes represent the same thing: that the oxygen measurement value is in units of mmHg. Both formats are used interchangeably within DCL’s because the only child element of the variable element is material amount. Therefore, units within the “variable” element will only refer to the units of the material amount value. By this reasoning, this difference can be discounted as an error in conversion. The same differences reported in DCL #241 also occurred in DCL #242; therefore, both of these two files are fully validated for file contents.

The remaining file with reported differences is DCL #240, with 98 differences between files. The vast majority of differences between the converted file and original file were due to elements and element attributes being ordered differently. Each of the 94 issues which fall under this category can be removed based on two reasons. The first reason that supports discarding these differences is that the converted DCL passed MCDS schema validation, so no elements or attributes appear out of the specified sequence. The second reason is that element and attribute values in the differently ordered sections are equivalent and each ordering difference occurs within the same parent element between files. For example, there are a total of eight variables with distinct name attributes used under the five microenvironment elements within each DCL#240 file. The contents of each variable element are equivalent when comparing each variable element underneath the microenvironment elements between files. The difference is the order in which the variable elements occur, and the order in which their attributes are organized. For example, the converted DCL always places the “oxygen” variable first, the “co2” variable second, and so forth within the microenvironment elements where these variable elements appear. This is due to the conversion to ISA-Tab, in which each distinctly named variable becomes an Assay header and is written in order of first appearance in the original DCL file. When the ISA-Tab file is converted back to a DCL, the script translates the data by column, resulting in the variables in the converted DCL being ordered by first appearance within the original DCL rather than in the same order as the original DCL. However, this is not a difference in file content because variables are not sequenced elements and can appear under a microenvironment in any ordering, as long as the content within the variable element is the same. Each of the 94 ordering differences can be attributed to this situation or a difference in attribute ordering within an element. As previously stated, each of these 94 differences are not considered to be issues since the file content of the converted DCL and original DCL match.

The remaining 4 differences between the converted and original versions of DCL #240 are attributed to differences in multiple occurrences of orcid-identifier elements for the same person within the original DCL. In the original DCL, an orcid-identifier element for “Kerri-Ann Norton” exists under the creator, current contact, and curator elements. However, each instance of the orcid-identifier element for this person holds different information in the text value of its child elements. For example, the current contact orcid-identifier lists the person’s organization as “Johns Hopkins” while the creator orcid-identifier lists the person’s organization as “Johns Hopkins University”. The orcid-identifier under the curator element does not contain any text values for this person’s email, organization name, or department name elements, whereas the other two orcid-identifiers contain this information. This difference is created as a result of the conversion script following ISA-Tab guidelines, which state that if a person has multiple roles within a Study, their roles should be joined by semicolon under one data field (see FIGURE 19).



```
STUDY CONTACTS
Study Person Last Name              "Norton"
Study Person First Name             "Kerri-Ann"
Study Person Mid Initials           ""
Study Person Email                  "kerri.norton@gmail.com"
Study Person Phone                  ""
Study Person Fax                    ""
Study Person Address                ""
Study Person Affiliation            "Johns Hopkins - Biomedical Engineering"
Study Person Roles                  "Current Contact;Curator;Creator"
```

FIGURE 19 – Differences in ORCiD ID Elements

The conversion script uses the information contained under the current contact element of the DCL being converted for the person’s information within the created Investigation file, as the current contact is the most relevant role listed in the DCL to contact with any questions regarding the file. The remaining hierarchy of roles by importance is curator, then creator, then last modified by; however, this ordering is only used when a person with multiple roles is not the current contact. Because each person is only represented by one set of identifying data fields in ISA-Tab, any difference in orcid-identifier data contained under role elements of lower rank will be lost. This is the only instance of true data loss by the conversion script. However, this is not of great detriment because the value within the path element is the person's ORCiD ID path. In any case where a person's email or affiliation is lost, the ORCiD ID path can be used to access the person's registered ORCiD ID which includes email address, affiliation, and other information. Furthermore, this is an issue with the input DCL rather than an issue with the conversion script. Multiple instances of orcid-identifier elements for the same person should be identical in content since each orcid-identifier instance is describing the same person. These four differences can also be discounted as conversion errors since they are issues with the input file instead of the conversion script.

Based on the prior discussion of differences between converted DCL's and original DCL's, all differences can be discounted as errors. Therefore, both the MCDS-DCL to ISA-Tab conversion script and the ISA-Tab to MCDS-DCL conversion script are fully verified for maintaining file contents for all existing DCL's.

C. Script Run Times

Run times for each conversion script along with the converted DCL evaluation script were documented to quantify script computing performance (see TABLE IV). The metrics for total run time and files processed per second are recorded values from within the tqdm status bar displayed in these scripts. The computer used to run these conversion scripts has 8GB of RAM and a 2.40GHZ processor, which is by no means a supercomputer. Majority of users who wish to run the conversion scripts or evaluation script can expect similar run times. Although the scripts were not created with optimal processing speed as a design factor, the quick run times allow for future conversions of a larger number of files in a reasonable amount of time. For example, if a user wanted to convert a collection of 2,000 MCDS-DCL files to ISA-Tab file sets, the conversion script would convert all files in approximately 14 minutes, assuming that the rate of files processed per second is the same as experienced in this project.

TABLE IV

CONVERSION AND EVALUATION SCRIPT RUN TIMES

Script Function	**Total Run Time (235 Input DCL's or ISA-Tab File Sets)**	**DCL's or ISA-Tab File Sets Processed per Second**
MCDS-DCL to ISA-Tab Conversion	98 seconds	2.4
ISA-Tab to MCDS-DCL Conversion	153 seconds	1.5
Validating and Verifying Converted DCL	8 seconds	29.4

IV. CONCLUSIONS

The created conversion scripts are demonstrated to meet the goals of this project, with the results of verification and validation testing supporting fully validated file conversion with no errors in data conversions. The MCDS-DCL to ISA-Tab conversion script is validated by the CSSI Metadata Utility software for the selected collection of ISA-Tab file sets, supporting this conversion script's validation for converting all of the 235 DCL's used as inputs. The ISA-Tab to MCDS-DCL conversion script is validated with the MCDS schema for all input ISA-Tab files, supporting the conversion script's validation for converting all 235 generated ISA-Tab files. The reported differences between converted DCL's and their counterpart original DCL's were analyzed to determine that no data losses or additions occurred that signify conversion error, supporting that both conversion scripts are verified for the 235 input DCL's.

As the library of DCL's expands, the conversion scripts and mapping of relationships between MCDS and ISA-Tab data fields are expected to require updating. The operation of the conversion scripts created in this project are reflective of this future need. A user that attempts conversion from MCDS-DCL to ISA-Tab can choose to search for data within the DCL which may not be supported for conversion. This allows a user to effortlessly determine if mapping relationships should be examined, or if the DCL will properly convert to ISA-Tab form. Providing mapping relationships within an Excel file is a format that is friendly to both user and computer. The Excel format is easy for users to understand the relationships between MCDS and ISA-Tab data fields as the relationships are provided in one compact document as opposed to being spread out among a conversion script. The Excel format is also beneficial to updating relationships for the script. Updates can be made to include MCDS elements or attributes for conversion that are not used in previous DCL's by simply typing the xPath of the element or attribute along with an appropriate name for the corresponding ISA-Tab label into a new row in the relationships Excel file. Since the script reads the entirety of each column in the relationships Excel file, the additional data fields will be imported into the conversion script without needing to change any code within the conversion script. If the added data fields follow a form which is already successfully handled by the script, there is a high probability of no changes needing to be made at all to the script to include the new elements. This will result in a reduced amount of time spent making future updates to the conversion scripts.

Additionally, the conversion scripts and data relationship mapping created in this project provide a starting point for converting any ISA-Tab file to a MCDS-DCL file. By modifying the data within an ISA-Tab file set to meet a format that can be handled by the ISA-Tab to MCDS-DCL conversion script, conversion of ISA-Tab files to DCL's that are not derived from MCDS-DCL files can be accomplished. This process can be completed by importing an ISA-Tab file set into the CSSI Metadata Utility software then altering the ISA labels and data to match a data structure prescribed in the relationship mapping Excel file. The CSSI Metadata Utility Software would be useful for this purpose as it provides a more user-friendly interface to edit an ISA-Tab file than using a text editor such as Notepad. Additionally, validation of the altered ISA-Tab document can be completed as the file is edited to ensure that changes do not violate ISA-Tab specifications.

As a whole, the work completed in this project yields vetted conversion scripts for transferring MCDS-DCL’s to ISA-Tab files and transferring ISA-Tab files in a specified format to MCDS-DCL’s. Furthermore, ISA-Tab files which do not follow the data structure used in the ISA-Tab to MCDS-DCL conversion script can now be converted to MCDS-DCL’s by first translating the file into a supported ISA-Tab conversion form. Finally, the findings of this project are a considerable building block to future efforts in converting MCDS Digital Snapshot files to ISA-Tab files and converting a wider input of ISA-Tab files to MCDS-DCL files in an automated manner.

V. RECOMMENDATIONS

The work completed in this project provides many views of ways that the MCDS project can be improved in the future. The first suggestion is to provide a "preferred format" of MCDS-DCL files. This format would not be mandated by the MCDS schema because the DCL framework must allow a degree of customization in order to express certain data. The preferred format would include standards for normalized spacing and tabbing between elements and a specification of elements which should contain identical content, such as orcid-identifier elements belonging to the same person. While providing documentation of the format would be beneficial, a parallel approach that would offer more utility is the creation of a Python script that updates an input DCL file to the preferred format for changes that do not alter data content and provides the user with an interactive way to select preferences for data content alterations, such as standardizing orcid-identifier elements belonging to the same person that are not identical. This script could be used when DCL's are created but before they are released, since a future update to remove a few blank lines does not warrant creation of an updated Digital Cell Line file. Although this script would make little change in the way that a computer views a Digital Cell Line, following a preferred format would improve readability for humans. As the MCDS project and the library of DCL's expands, this preferred format would help to maintain a degree of standardization among curated DCL's. Additionally, many features that may be desired for a preferred formatting script are already included within the ISA-Tab to MCDS-DCL conversion script. Sections of the conversion script such as the function that properly indents an Element Tree could easily be transferred to a preferred formatting script to decrease the effort needed to accomplish this goal.

A second suggestion for the future is to create a converted DCL evaluation script that uses the MCDS schema to determine whether differences in ordering matter or not. During this project, an XML canonicalization module was used in an attempt to decrease output differences that are not indicative of conversion errors and instead were a difference in ordering of elements or spacing. While canonical form does standardize the XML file, this did not work as expected because element ordering is unchanged and spacing between alphanumeric characters is kept when transferring to canonical form. On the other hand, an XML diff script that utilizes the MCDS schema would be able to determine whether sequencing matters for an element and not considering ordering differences for non-sequenced elements as errors. Additionally, the proposed preferred formatting could eliminate the spacing differences within data values being compared. The goal of this process would be to create an evaluation script than can truly verify a converted MCDS-DCL file against an original MCDS-DCL file for data contents rather than only provide differences. Such an avenue was not pursued during this project because there were few differences between converted DCL files and their counterpart original DCL files, so the benefits in a more automated verification process offered little return on time invested. In the future, this would be beneficial if a large number of DCL files that require updates to mapping relationships are converted to ISA-Tab format as it would allow a more succinct verification of converted file contents.

A third suggestion involves creating a more robust conversion system before new DCL's require updating to mapping relationships and the conversion scripts. The mapping relationships established in this project were limited in scope to the elements and attributes within the set of existing DCL files. However, all established elements and attributes (i.e., not "custom" elements) for MCDS are specified within the MCDS schema files. Many of these elements and attributes will only exist in MCDS Digital Snapshot files, but others might appear in future DCL's as the library expands. Elements and attributes could be mapped within the relationships Excel file and tested for conversion before they appear in a future DCL. An XML file containing each element and attribute within the MCDS schema was generated for this project as one avenue to create a DCL template XML for converting ISA-Tab files to a MCDS-DCL file (TABLE VII, "Full_fromSchema.xml"). This file could be utilized to test file conversion for new elements or attributes before a DCL with data fields which are not supported by current relationship mapping is created. Following this method to increase the number of MCDS elements supported by relationship mapping and the conversion scripts would help streamline the process of converting new DCL's in the future.

The final suggestion for future work is an idea for improving the ISA-Tab to MCDS-DCL script. A desired function of the ISA-Tab to MCDS-DCL script is to be able to parse through a set of ISA-Tab files in order to generate an MCDS-DCL file without needing the ISA-Tab files to be of a strict form. This is a challenge because headers/labels in Study and Assay files are largely customized by the creator of the file. Therefore, it is unrealistic to match label text to determine what the data within a Study or Assay file correlates to within the MCDS-DCL framework when using ISA-Tab files created independent from this project as conversion inputs. The functionality of the ISA-Tab to MCDS-DCL conversion script could be improved with regards to accepting a wider amount of input data by mapping of IDs such as ChEBI ID and UniProt ID used in MCDS-DCL files with Term Source REFs used in ISA-Tab files. This would allow the conversion script to match ISA-Tab data to correlate MCDS-DCL data based on a standardized ontology rather than needing a textual match to an ISA-Tab label. Mapping could be accomplished either by creating a local file containing Term Source REFs, IDs used within MCDS-DCL files, and the desired location within the MCDS-DCL to write data, or a web scraping script that searches for Term Source REFs and IDs during execution of the conversion script as identification data fields appear. The user of the script could then be prompted to choose a location for data within the ISA-Tab file which has not been converted to the MCDS-DCL file or choose to not include the data for conversion. This approach might require acceptance of multiple ISA-Tab file sets to the conversion script in order to amass the information contained in one MCDS-DCL.

REFERENCES CITED

Friedman, Samuel. “MultiCellDS User Overview.” *Figshare*, 30 November 2016, available from https://doi.org/10.6084/m9.figshare.4269254.v1; accessed 12 April 2021

Friedman, Samuel. “MultiCellDS Version 1.0.0 Full Documentation.” *Figshare*, 30 November 2016, available from https://doi.org/10.6084/m9.figshare.4269269.v1; accessed 12 April 2021

Friedman, Samuel H., et al. “MultiCellDS: A Community-Developed Standard for Curating Microenvironment-Dependent Multicellular Data.” *bioRxiv*. Cold Spring Harbor Laboratory, 1 January 2016, available from https://www.biorxiv.org/content/10.1101/090456v1.full; accessed 12 April 2021.

Ghaffarizadeh, Ahmadreza, et al. “PhysiCell: An Open Source Physics-Based Cell Simulator for 3-D Multicellular Systems.” *PLOS Computational Biology*. Public Library of Science, available from https://journals.plos.org/ploscompbiol/article?id=10.1371%2Fjournal.pcbi.1005991; accessed 12 April 2021.

“GigaDB Datasets.” *GigaDB*, available from http://gigadb.org/; accessed 12 April 2021.

“ISA Abstract Model.” *ISA Abstract Model - ISA Model and Serialization Specifications 1.0 documentation*, available from https://isa-specs.readthedocs.io/en/latest/isamodel.html; accessed 12 April 2021.

“ISA-Tab Format.” *ISA-Tab format - ISA Model and Serialization Specifications 1.0 documentation*, available from https://isa-specs.readthedocs.io/en/latest/isatab.html; accessed 12 April 2021.

Macklin, Paul. “Key Challenges Facing Data-Driven Multicellular Systems Biology.” *GigaScience*, Oxford University Press, 1 October 2019, available from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6812467/; accessed 12 April 2021.

——. “The MultiCellular Data Standard Project.” *MultiCellDS,* available from http://multicellds.org/; accessed 12 April 2021.

Rocca-Serra, Philippe, et al. “ISA Software Suite: Supporting Standards-Compliant Experimental Annotation and Enabling Curation at the Community Level.” *OUP Academic,* Oxford Academic, 2 August 2010, available from https://academic.oup.com/bioinformatics/article/26/18/2354/208338?login=true; accessed 12 April 2021.

“Validating Metadata - CSSI Data Coordinating Center - NCI Wiki.” *National Institutes of Health.* U.S. Department of Health and Human Services, available from https://wiki.nci.nih.gov/display/DSE/Validating+Metadata; accessed 12 April 2021.

Wake, Marvalee H. “Integrative Biology: Science for the 21st Century.” *OUP Academic*, Oxford University Press, 1 April 2008, available from https://academic.oup.com/bioscience/article/58/4/349/310332?login=true; accessed 12 April 2021.

Wellmann, Janina. “Gluing Life Together. Computer Simulation in the Life Sciences: An Introduction,” *NCBI*, 22 November 2018, available from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6267125/; accessed 12 April 2021.

Wolkenhauer, Olaf, and Allan Muir. “The Complexity of Cell-Biological Systems.” *Philosophy of Complex Systems*, North-Holland, 3 June 2011, available from https://www.sciencedirect.com/science/article/pii/B9780444520760500134; accessed 12 April 2021.

“XML Elements”, *W3Schools*, available from https://www.w3schools.com/xml

APPENDIX I.

PROJECT SOFTWARE AND FILES

TABLE V

SOFTWARES UTILIZED IN PROJECT

Software Name	Version Used
Microsoft Excel 2016	2102
Notepad++	7.8.6
Python	3.9
PyCharm (Community Edition)	2020.2.3
CSSI Metadata Utility	N/A

TABLE VI

PYTHON LIBRARIES AND MODULES UTILIZED IN PROJECT

Library	Module(s)	Version	Documentation Link
Python Standard Library	os, sys, copy, argparse, re	3.9	https://docs.python.org/3/library/
lxml	Etree	4.6.2	https://lxml.de/index.html#documentation
pandas	ExcelWriter, DataFrame, read_excel, read_csv	1.2.1	https://pandas.pydata.org/docs/
tqdm	Tqdm	4.59.0	https://tqdm.github.io/
xmldiff	main, formatting	2.4	https://pypi.org/project/xmldiff/
xmlschema	XMLSchema	1.6.0	https://pypi.org/project/xmlschema/
pigar	N/A	0.10.0	https://pypi.org/project/pigar/

TABLE VII

PROJECT FILES AVAILABLE ON GITHUB

File or Directory Name	Purpose/Content	Web Link
mcds_content_list.py	Using an input of MCDS-DCL files, generate an Excel file that containing information about the data elements within the input XML file(s)	https://github.com/rheiland/mcds2isa/blob/DCL_ISA_Conversion_v1.0/MCDS_DCL2ISATab/mcds_content_list.py
MCDS_DCL_All_Content.xlsx	Generated by the mcds_content_list.py script. Contains all elements and attributes contained in existing DCL files, in addition to element xPaths and classification of the element or attribute	https://github.com/rheiland/mcds2isa/blob/DCL_ISA_Conversion_v1.0/MCDS_DCL2ISATab/MCDS_DCL_All_Content.xlsx
MCDS_DCL_DataLog.xlsx	Initially copied from the MCDS_DCL_All_Content.xlsx file. Contains analysis performed on DCL content. Also contains a log of data element usage in conversion scripts and location of conversion operation for the data element within the conversion script	https://github.com/rheiland/mcds2isa/blob/DCL_ISA_Conversion_v1.0/MCDS_DCL2ISATab/MCDS_DCL_DataLog.xlsx
MultiCellDS.xsd	XML schema documented of supported MultiCellDS data elements. To use the schema, all associated schema (.xsd) files within the v1.0.0 folder must be in the same folder as the MultiCellDS.xsd schema file	https://gitlab.com/MultiCellDS/MultiCellDS/-/tree/transitions/v1.0/v1.0.0**

File or Directory Name	Purpose/Content	Web Link
DCL_xml Updates.xlsx	Documents reason for creation of new DCL's from existing DCL's, the name of the new file, and the name of the parent file	https://github.com/rheiland/mcds2isa/blob/DCL_ISA_Conversion_v1.0/MCDS_DCL2ISATab/DCL_xml%20Updates.xlsx
ISA_MCDS_DCL_Relationships.xlsx	Contains MCDS-DCL and ISA-Tab relationship mapping and information to guide script decisions, utilized by conversion scripts to create converted file outputs	https://github.com/rheiland/mcds2isa/blob/DCL_ISA_Conversion_v1.0/MCDS_DCL2ISATab/ISA_MCDS_DCL_Relationships.xlsx
MCDS_DCL2ISA_DB.py	Convert an input MCDS-DCL XML file to a set of ISA-Tab Investigation, Study, and Assay files	https://github.com/rheiland/mcds2isa/blob/DCL_ISA_Conversion_v1.0/MCDS_DCL2ISATab/MCDS_DCL2ISA_DB.py
All_Digital_Cell_Lines	Folder containing all MCDS-DCL files, including those created in updates	https://github.com/rheiland/mcds2isa/tree/DCL_ISA_Conversion_v1.0/All_Digital_Cell_Lines
ISATabOutput	Folder containing all ISA-Tab file sets created from the MCDS-DCL to ISA-Tab conversion script	https://github.com/rheiland/mcds2isa/tree/DCL_ISA_Conversion_v1.0/MCDS_DCL2ISATab/ISATabOutput
DCL_CleanTemplate.xml	XML file containing all element tags and attributes which are used within existing DCL files, with all element text and attribute values set to "". This file is used when converting an ISA-Tab file set to a MCDS-DCL file	https://github.com/rheiland/mcds2isa/blob/DCL_ISA_Conversion_v1.0/ISATab2MCDS_DCL/DCL%20Template/DCL_CleanTemplate.xml

File or Directory Name	Purpose/Content	Web Link
ISA_2_MCDS_DCL.py	Script that converts an input ISA-Tab file set to a MCDS-DCL XML file	https://github.com/rheiland/mcds2isa/blob/DCL_ISA_Conversion_v1.0/ISATab2MCDS_DCL/ISA_2_MCDS_DCL.py
MCDS Conversion Output	Folder containing all converted DCL files that were generated from the ISA-Tab to MCDS-DCL conversion script	https://github.com/rheiland/mcds2isa/tree/DCL_ISA_Conversion_v1.0/ISATab2MCDS_DCL/MCDS%20Conversion%20Output
convertedDCL_eval.py	Script that outputs and Excel containing results of validation and file content verification for converted DCL files	https://github.com/rheiland/mcds2isa/blob/DCL_ISA_Conversion_v1.0/ISATab2MCDS_DCL/convertedDCL_eval.py
DCLConversionEval.xlsx	Excel file containing validation and file content verification testing results, generated by convertedDCL_eval.py	https://github.com/rheiland/mcds2isa/blob/DCL_ISA_Conversion_v1.0/ISATab2MCDS_DCL/DCLConversionEval.xlsx
requirements.txt	Contains text file of external Python libraries used in the project, versions of the libraries, and what files they are used in. This file can be used to install the specified libraries for easier setup for use of scripts from the project	https://github.com/rheiland/mcds2isa/blob/DCL_ISA_Conversion_v1.0/MCDS_DCL2ISATab/requirements.txt
Full_fromSchema.xml	XML file containing all element tag and attributes which are supported by MCDS	https://github.com/rheiland/mcds2isa/blob/DCL_ISA_Conversion_v1.0/ISATab2MCDS_DCL/DCL%20Template/Full_fromSchema.xml

File or Directory Name	Purpose/Content	Web Link
MCDS_DCL_InputCharacteristics.py	Python Script used to create a text file containing MCDS-DCL file names, their index within the All Digital Cell Lines directory, and any issues related to file type attribute or ID attribute ordering within elements	https://github.com/rheiland/mcds2isa/blob/DCL_ISA_Conversion_v1.0/MCDS_DCL2ISATab/MCDS_DCL_InputCharacteristics.py
mcds_dcl2isa.py	The old MCDS-DCL to ISA-Tab conversion script, updated to run in Python 3	https://github.com/rheiland/mcds2isa/blob/DCL_ISA_Conversion_v1.0/mcds_dcl2isa.py

General Note – at the time of writing, all documents in TABLE III without asterisks are located in the “DCL_ISA_Conversion_v1.0” branch within the “mcds2isa” repository of user “rheiland” on GitHub. This branch is likely to be merged with the master branch in the future.

*hosted on GitLab, not the GitHub repository

**at the time of writing, the updated MCDS schema is housed within the transitions branch: this will likely be merged with the master branch in the future

APPENDIX II.

SECTION OF DATALOG EXCEL FILE

TABLE VIII

EXAMPLES OF DCL FILE CONTENT FROM EXCEL

MCDS Entity	xPath - Index Removed	File Name	Entity Type	Multiple or Single	Total Entity Occurrences	Total xPath Occurrences	Discrete File Entity Occurrences	Discrete File xPath Occurrences
version	.[@version]	MCDS_L_0000000003.xml	Attribute	Single	470	235	470	235
cell_line	cell_line	MCDS_L_0000000003.xml	Attribute Element	Single	235	235	235	235
ID	cell_line[@ID]	MCDS_L_0000000003.xml	Attribute	Single	3187	235	1710	235
label	cell_line[@label]	MCDS_L_0000000003.xml	Attribute	Single	235	235	235	235
ID	cell_line/metadata/MultiCellDB/ID	MCDS_L_0000000003.xml	Text Element	Single	3187	235	1710	235
name	cell_line/metadata/MultiCellDB/name	MCDS_L_0000000003.xml	Text Element	Single	2225	235	1120	235
description	cell_line/metadata/description	MCDS_L_0000000003.xml	Text Element	Single	235	235	235	235
text	cell_line/metadata/citation/text	MCDS_L_0000000003.xml	Text Element	Single	238	235	238	238
URL	cell_line/metadata/citation/URL	MCDS_L_0000000003.xml	Text Element	Single	1424	63	1178	63

MCDS Entity	xPath - Index Removed	File Name	Entity Type	Multiple or Single	Total Entity Occurrences	Total xPath Occurrences	Discrete File Entity Occurrences	Discrete File xPath Occurrences
curation	cell_line/metadata/curation	MCDS_L_0000000003.xml	Attribute Element	Single	236	235	233	233
curated	cell_line/metadata/curation[@curated]	MCDS_L_0000000003.xml	Attribute	Single	233	233	233	233
created	cell_line/metadata/curation/created	MCDS_L_0000000003.xml	Text Element	Single	235	235	235	235
last_modified	cell_line/metadata/curation/last_modified	MCDS_L_0000000003.xml	Text Element	Single	235	235	235	235

APPENDIX III.

RESULTS OF CONVERTED DCL TESTS

TABLE IX

CONVERTED DCL FULL TESTING RESULTS

Testing Index	DCL Filename	Passed Validation	Total Data Entities in Converted File	Num of Issues in Converted File
1	MCDS_L_0000000003	TRUE	128	0
2	MCDS_L_0000000004	TRUE	128	0
3	MCDS_L_0000000007	TRUE	128	0
4	MCDS_L_0000000008	TRUE	128	0
5	MCDS_L_0000000009	TRUE	128	0
6	MCDS_L_0000000010	TRUE	128	0
7	MCDS_L_0000000011	TRUE	128	0
8	MCDS_L_0000000012	TRUE	128	0
9	MCDS_L_0000000013	TRUE	128	0
10	MCDS_L_0000000014	TRUE	128	0
11	MCDS_L_0000000015	TRUE	128	0
12	MCDS_L_0000000016	TRUE	128	0
13	MCDS_L_0000000017	TRUE	128	0
14	MCDS_L_0000000018	TRUE	128	0

Testing Index	DCL Filename	Passed Validation	Total Data Entities in Converted File	Num of Issues in Converted File
15	MCDS_L_0000000019	TRUE	128	0
16	MCDS_L_0000000020	TRUE	128	0
17	MCDS_L_0000000021	TRUE	128	0
18	MCDS_L_0000000022	TRUE	128	0
19	MCDS_L_0000000023	TRUE	128	0
20	MCDS_L_0000000024	TRUE	128	0
21	MCDS_L_0000000025	TRUE	128	0
22	MCDS_L_0000000026	TRUE	128	0
23	MCDS_L_0000000027	TRUE	128	0
24	MCDS_L_0000000028	TRUE	128	0
25	MCDS_L_0000000029	TRUE	128	0
26	MCDS_L_0000000030	TRUE	128	0
27	MCDS_L_0000000031	TRUE	128	0
28	MCDS_L_0000000032	TRUE	128	0
29	MCDS_L_0000000033	TRUE	128	0
30	MCDS_L_0000000034	TRUE	128	0
31	MCDS_L_0000000035	TRUE	128	0
32	MCDS_L_0000000036	TRUE	128	0
33	MCDS_L_0000000037	TRUE	128	0
34	MCDS_L_0000000038	TRUE	128	0
35	MCDS_L_0000000039	TRUE	128	0
36	MCDS_L_0000000040	TRUE	128	0
37	MCDS_L_0000000041	TRUE	128	0

Testing Index	DCL Filename	Passed Validation	Total Data Entities in Converted File	Num of Issues in Converted File
38	MCDS_L_0000000042	TRUE	128	0
39	MCDS_L_0000000044	TRUE	162	2
40	MCDS_L_0000000047	TRUE	162	2
41	MCDS_L_0000000048	TRUE	162	2
42	MCDS_L_0000000049	TRUE	736	0
43	MCDS_L_0000000050	TRUE	736	0
44	MCDS_L_0000000051	TRUE	736	0
45	MCDS_L_0000000052	TRUE	736	0
46	MCDS_L_0000000053	TRUE	736	0
47	MCDS_L_0000000054	TRUE	736	0
48	MCDS_L_0000000055	TRUE	736	0
49	MCDS_L_0000000056	TRUE	736	0
50	MCDS_L_0000000057	TRUE	736	0
51	MCDS_L_0000000058	TRUE	736	0
52	MCDS_L_0000000059	TRUE	736	0
53	MCDS_L_0000000060	TRUE	736	0
54	MCDS_L_0000000061	TRUE	711	0
55	MCDS_L_0000000062	TRUE	711	0
56	MCDS_L_0000000063	TRUE	711	0
57	MCDS_L_0000000064	TRUE	711	0
58	MCDS_L_0000000065	TRUE	711	0
59	MCDS_L_0000000066	TRUE	179	0
60	MCDS_L_0000000067	TRUE	179	0

Testing Index	DCL Filename	Passed Validation	Total Data Entities in Converted File	Num of Issues in Converted File
61	MCDS_L_0000000068	TRUE	179	0
62	MCDS_L_0000000069	TRUE	179	0
63	MCDS_L_0000000070	TRUE	178	0
64	MCDS_L_0000000071	TRUE	179	0
65	MCDS_L_0000000072	TRUE	179	0
66	MCDS_L_0000000073	TRUE	179	0
67	MCDS_L_0000000074	TRUE	179	0
68	MCDS_L_0000000075	TRUE	179	0
69	MCDS_L_0000000076	TRUE	179	0
70	MCDS_L_0000000077	TRUE	179	0
71	MCDS_L_0000000078	TRUE	179	0
72	MCDS_L_0000000079	TRUE	179	0
73	MCDS_L_0000000080	TRUE	179	0
74	MCDS_L_0000000081	TRUE	179	0
75	MCDS_L_0000000082	TRUE	179	0
76	MCDS_L_0000000083	TRUE	179	0
77	MCDS_L_0000000084	TRUE	179	0
78	MCDS_L_0000000085	TRUE	179	0
79	MCDS_L_0000000086	TRUE	179	0
80	MCDS_L_0000000087	TRUE	179	0
81	MCDS_L_0000000088	TRUE	179	0
82	MCDS_L_0000000089	TRUE	179	0
83	MCDS_L_0000000090	TRUE	179	0

Testing Index	DCL Filename	Passed Validation	Total Data Entities in Converted File	Num of Issues in Converted File
84	MCDS_L_0000000091	TRUE	179	0
85	MCDS_L_0000000092	TRUE	179	0
86	MCDS_L_0000000093	TRUE	179	0
87	MCDS_L_0000000094	TRUE	179	0
88	MCDS_L_0000000095	TRUE	179	0
89	MCDS_L_0000000096	TRUE	179	0
90	MCDS_L_0000000097	TRUE	179	0
91	MCDS_L_0000000098	TRUE	179	0
92	MCDS_L_0000000099	TRUE	179	0
93	MCDS_L_0000000100	TRUE	179	0
94	MCDS_L_0000000101	TRUE	179	0
95	MCDS_L_0000000102	TRUE	179	0
96	MCDS_L_0000000103	TRUE	179	0
97	MCDS_L_0000000104	TRUE	179	0
98	MCDS_L_0000000105	TRUE	173	0
99	MCDS_L_0000000106	TRUE	173	0
100	MCDS_L_0000000107	TRUE	173	0
101	MCDS_L_0000000108	TRUE	173	0
102	MCDS_L_0000000109	TRUE	173	0
103	MCDS_L_0000000110	TRUE	172	0
104	MCDS_L_0000000111	TRUE	173	0
105	MCDS_L_0000000112	TRUE	173	0
106	MCDS_L_0000000113	TRUE	173	0

Testing Index	**DCL Filename**	**Passed Validation**	**Total Data Entities in Converted File**	**Num of Issues in Converted File**
107	MCDS_L_0000000114	TRUE	172	0
108	MCDS_L_0000000115	TRUE	173	0
109	MCDS_L_0000000116	TRUE	173	0
110	MCDS_L_0000000117	TRUE	173	0
111	MCDS_L_0000000118	TRUE	173	0
112	MCDS_L_0000000119	TRUE	173	0
113	MCDS_L_0000000120	TRUE	173	0
114	MCDS_L_0000000121	TRUE	173	0
115	MCDS_L_0000000122	TRUE	173	0
116	MCDS_L_0000000123	TRUE	173	0
117	MCDS_L_0000000124	TRUE	173	0
118	MCDS_L_0000000125	TRUE	173	0
119	MCDS_L_0000000126	TRUE	173	0
120	MCDS_L_0000000127	TRUE	173	0
121	MCDS_L_0000000128	TRUE	173	0
122	MCDS_L_0000000129	TRUE	173	0
123	MCDS_L_0000000130	TRUE	173	0
124	MCDS_L_0000000131	TRUE	173	0
125	MCDS_L_0000000132	TRUE	173	0
126	MCDS_L_0000000133	TRUE	173	0
127	MCDS_L_0000000134	TRUE	173	0
128	MCDS_L_0000000135	TRUE	173	0
129	MCDS_L_0000000136	TRUE	173	0

Testing Index	DCL Filename	Passed Validation	Total Data Entities in Converted File	Num of Issues in Converted File
130	MCDS_L_0000000137	TRUE	173	0
131	MCDS_L_0000000138	TRUE	173	0
132	MCDS_L_0000000139	TRUE	173	0
133	MCDS_L_0000000140	TRUE	173	0
134	MCDS_L_0000000141	TRUE	173	0
135	MCDS_L_0000000142	TRUE	173	0
136	MCDS_L_0000000143	TRUE	173	0
137	MCDS_L_0000000144	TRUE	173	0
138	MCDS_L_0000000145	TRUE	173	0
139	MCDS_L_0000000146	TRUE	173	0
140	MCDS_L_0000000147	TRUE	173	0
141	MCDS_L_0000000148	TRUE	173	0
142	MCDS_L_0000000149	TRUE	173	0
143	MCDS_L_0000000150	TRUE	173	0
144	MCDS_L_0000000151	TRUE	173	0
145	MCDS_L_0000000152	TRUE	173	0
146	MCDS_L_0000000153	TRUE	173	0
147	MCDS_L_0000000154	TRUE	173	0
148	MCDS_L_0000000155	TRUE	173	0
149	MCDS_L_0000000156	TRUE	173	0
150	MCDS_L_0000000157	TRUE	173	0
151	MCDS_L_0000000158	TRUE	173	0
152	MCDS_L_0000000159	TRUE	173	0

Testing Index	**DCL Filename**	**Passed Validation**	**Total Data Entities in Converted File**	**Num of Issues in Converted File**
153	MCDS_L_0000000160	TRUE	173	0
154	MCDS_L_0000000161	TRUE	173	0
155	MCDS_L_0000000162	TRUE	173	0
156	MCDS_L_0000000163	TRUE	173	0
157	MCDS_L_0000000164	TRUE	173	0
158	MCDS_L_0000000165	TRUE	173	0
159	MCDS_L_0000000166	TRUE	173	0
160	MCDS_L_0000000167	TRUE	173	0
161	MCDS_L_0000000168	TRUE	173	0
162	MCDS_L_0000000169	TRUE	173	0
163	MCDS_L_0000000170	TRUE	173	0
164	MCDS_L_0000000171	TRUE	173	0
165	MCDS_L_0000000172	TRUE	173	0
166	MCDS_L_0000000173	TRUE	173	0
167	MCDS_L_0000000174	TRUE	173	0
168	MCDS_L_0000000175	TRUE	173	0
169	MCDS_L_0000000176	TRUE	173	0
170	MCDS_L_0000000177	TRUE	173	0
171	MCDS_L_0000000178	TRUE	173	0
172	MCDS_L_0000000179	TRUE	173	0
173	MCDS_L_0000000180	TRUE	173	0
174	MCDS_L_0000000181	TRUE	173	0
175	MCDS_L_0000000182	TRUE	173	0

Testing Index	DCL Filename	Passed Validation	Total Data Entities in Converted File	Num of Issues in Converted File
176	MCDS_L_0000000183	TRUE	173	0
177	MCDS_L_0000000184	TRUE	173	0
178	MCDS_L_0000000185	TRUE	173	0
179	MCDS_L_0000000186	TRUE	173	0
180	MCDS_L_0000000187	TRUE	173	0
181	MCDS_L_0000000188	TRUE	173	0
182	MCDS_L_0000000189	TRUE	173	0
183	MCDS_L_0000000190	TRUE	173	0
184	MCDS_L_0000000191	TRUE	173	0
185	MCDS_L_0000000192	TRUE	173	0
186	MCDS_L_0000000193	TRUE	173	0
187	MCDS_L_0000000194	TRUE	173	0
188	MCDS_L_0000000195	TRUE	173	0
189	MCDS_L_0000000196	TRUE	173	0
190	MCDS_L_0000000197	TRUE	173	0
191	MCDS_L_0000000198	TRUE	173	0
192	MCDS_L_0000000199	TRUE	173	0
193	MCDS_L_0000000200	TRUE	173	0
194	MCDS_L_0000000201	TRUE	173	0
195	MCDS_L_0000000202	TRUE	173	0
196	MCDS_L_0000000203	TRUE	173	0
197	MCDS_L_0000000204	TRUE	173	0
198	MCDS_L_0000000205	TRUE	173	0

Testing Index	DCL Filename	Passed Validation	Total Data Entities in Converted File	Num of Issues in Converted File
199	MCDS_L_0000000206	TRUE	173	0
200	MCDS_L_0000000207	TRUE	173	0
201	MCDS_L_0000000208	TRUE	173	0
202	MCDS_L_0000000209	TRUE	173	0
203	MCDS_L_0000000210	TRUE	173	0
204	MCDS_L_0000000211	TRUE	173	0
205	MCDS_L_0000000212	TRUE	173	0
206	MCDS_L_0000000213	TRUE	173	0
207	MCDS_L_0000000214	TRUE	173	0
208	MCDS_L_0000000215	TRUE	173	0
209	MCDS_L_0000000216	TRUE	173	0
210	MCDS_L_0000000217	TRUE	173	0
211	MCDS_L_0000000218	TRUE	173	0
212	MCDS_L_0000000219	TRUE	173	0
213	MCDS_L_0000000220	TRUE	173	0
214	MCDS_L_0000000221	TRUE	173	0
215	MCDS_L_0000000222	TRUE	173	0
216	MCDS_L_0000000223	TRUE	173	0
217	MCDS_L_0000000224	TRUE	172	0
218	MCDS_L_0000000225	TRUE	172	0
219	MCDS_L_0000000226	TRUE	172	0
220	MCDS_L_0000000227	TRUE	172	0
221	MCDS_L_0000000228	TRUE	172	0

Testing Index	DCL Filename	Passed Validation	Total Data Entities in Converted File	Num of Issues in Converted File
222	MCDS_L_0000000229	TRUE	172	0
223	MCDS_L_0000000230	TRUE	172	0
224	MCDS_L_0000000231	TRUE	172	0
225	MCDS_L_0000000232	TRUE	172	0
226	MCDS_L_0000000233	TRUE	172	0
227	MCDS_L_0000000234	TRUE	172	0
228	MCDS_L_0000000235	TRUE	172	0
229	MCDS_L_0000000236	TRUE	172	0
230	MCDS_L_0000000237	TRUE	172	0
231	MCDS_L_0000000238	TRUE	176	0
232	MCDS_L_0000000239	TRUE	176	0
233	MCDS_L_0000000240	TRUE	536	98
234	MCDS_L_0000000241	TRUE	102	5
235	MCDS_L_0000000242	TRUE	97	5

VITA

Connor Joseph Burns graduated with his B.S. in Bioengineering from the University of Louisville in May 2020 and his M.Eng. in Bioengineering from the University of Louisville in May 2021. During his collegiate studies, he was recognized as a Grawemeyer Scholar and a recipient of the Steven Vanover Memorial scholarship. Connor's previous works include an independent study to build a 3D printed animatronic hand under the mentorship of Dr. Jonathan Kopechek. Connor plans to apply his passion for problem solving, learning, and a multi-disciplinary skill set to a Biomedical Engineering career in the future.