

Geological Map Production Toolbox v1 Report

1 CONTENTS

2	User guide	2
2.1	Overview	2
2.2	How to use Geological map production toolbox (In initial map production mode).....	2
2.2.1	Preparing for digitising.....	2
2.2.2	Run script: “1 Create Template Geological Features “	2
2.2.3	Digitise geological contacts and map boundary	3
2.2.4	Run Script: “2 Create Geological Unit Polygons”	4
2.2.5	Set unit_code for Geological Units	5
2.3	How to use Geological map production toolbox (Update Geological Map mode).....	6
2.3.1	“3 Update Geological Unit Polygons” Script	6
2.3.2	Editing scenario 1: Moving a contact or changing a contact type	7
2.3.3	Editing scenario 2: Adding a new contact in	8
2.4	Future additions	8
2	Programmer’s Guide	8
2.5	Overview	8
2.6	Toolbox functions overview	9
2.6.1	Initial map production mode	9
	1 Create Template Geological Features	9
2.6.2	Update Geological Map mode uses	9
2.7	Scripts explained	10
2.7.1	Script style.....	10
2.7.2	Code sources	10
2.7.3	Script format	10
2.7.4	Modules used.....	10
2.7.5	Naming Scheme for variables	11
2.7.6	Environment setting, path concatenation	11
2.7.7	Defaults used in scripts	11
2.8	Script code features	12
2.8.1	Digitising accuracy (script 2 and 3)	12
2.8.2	Increment trailing numbers function from script 3.	12

2 USER GUIDE

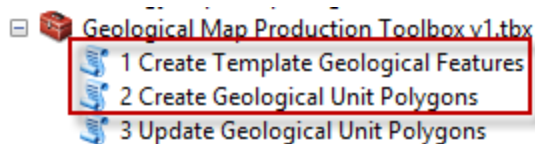
2.1 OVERVIEW

This user guide is written for users with a familiarity with basic operation of ArcGIS desktop including: georeferencing and editing. Numerous tutorials are available for these activities online.

There are two ways to use the Geological Map Production toolbox. 1) In initial production mode, 2) in Update Geological Map mode. These will be outlined below.

2.2 HOW TO USE GEOLOGICAL MAP PRODUCTION TOOLBOX (IN INITIAL MAP PRODUCTION MODE)

There are two scripts for initial map production: “**1 Create Template Geological Features**” and “**2 Create Geological Unit Polygons**”.



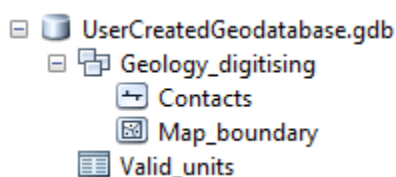
The steps required by the user to use these are outlined below.

2.2.1 Preparing for digitising

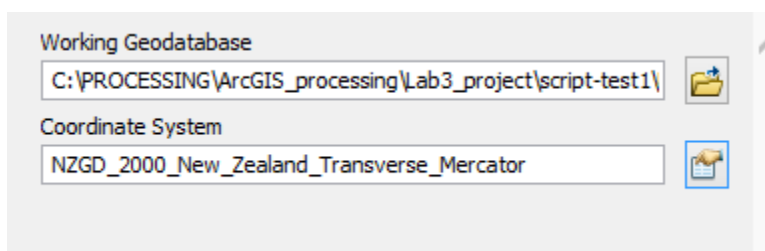
1. Scan and georeference your field map
2. Create a file geodatabase to store the features classes for digitising

2.2.2 Run script: “1 Create Template Geological Features “

A feature dataset is created inside the file geodatabase called “**Geology_digitising**” and empty feature classes for contacts and map boundary are created within this feature dataset. An empty table called Valid Units is created with fields for “**unit_code**” and “**Unit_name.**”



The valid units table has field for “**unit_code**” and “**Unit_name.**”



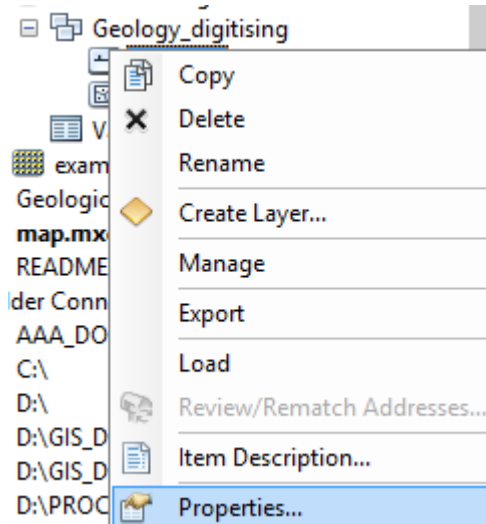
Working Geodatabase: This is the Geodatabase where the template geological features will be created.

Coordinate System: Select the coordinate system you would like to use for digitising.

2.2.3 Set contact sub-types for digitising

By default sub-types will be created for drawing accurate, approximate and inferred contacts. If you desire other contact types you can set them.

1. Right click on the Contacts layer and choose Properties



2. GO to the Sub-types tab.
3. Enter in additional sub-types by typing a unique number and the name.

Subtypes:

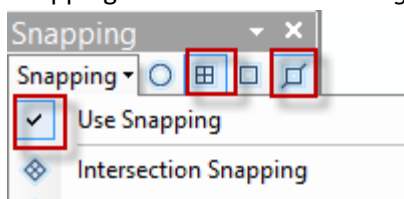
	Code	Description
	1	Accurate
	2	Approximate
	3	Inferred
	4	Fault
	5	Fault concealed

Default Values and Domains:

4. Save and exit.
5. Adding or re-adding the layer to your map will show the additional sub-types.

2.2.4 Digitise geological contacts and map boundary

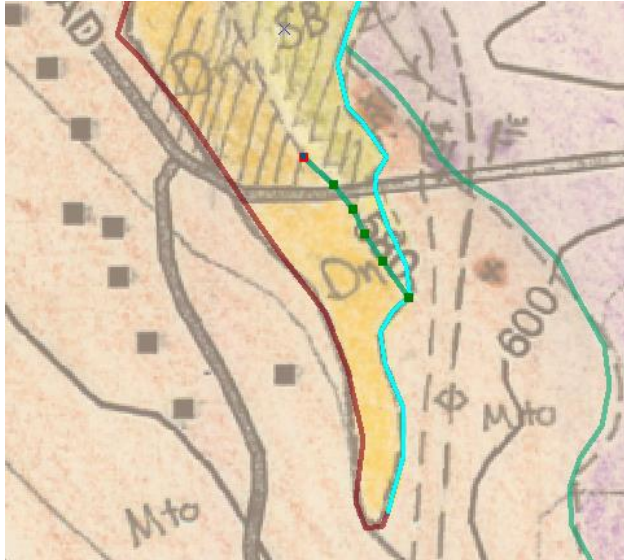
1. Add the **Geological_digitising** feature dataset to your map to get both **Contacts** and **Map_boundary** in for editing.
2. Recommended settings prior to editing:
 - 2.1. **Contracts** and **Map_boundary**: 40% Transparency
 - 2.2. Line width: 2
 - 2.3. Snapping Turned on: *End and Edge*



- 2.4. All other layers that could interfere with snapping turned off

Note: Snapping is not required for the scripts since the scripts include a snapping routine based on field map scale/digitising accuracy.

3. Start editing session
4. Draw in your **Map_boundaries**
5. Draw in your **Contacts**. Start a new line wherever your mapping accuracy changes, see red vs. green lines in this image:



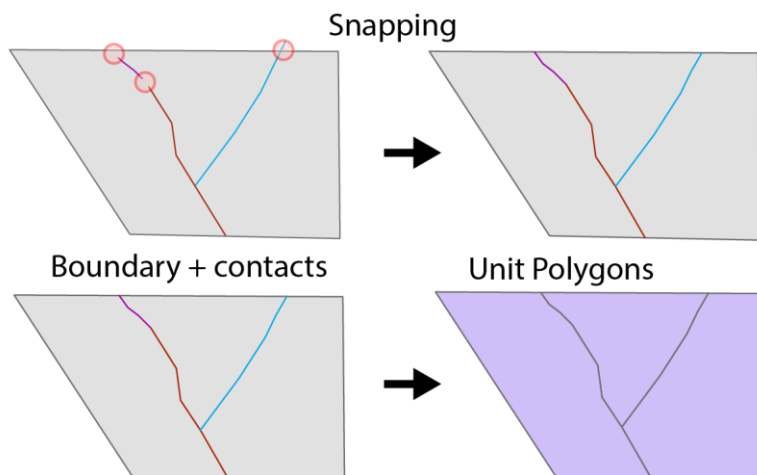
6. Populate the **Valid_Units** table with the geological units that you have mapped

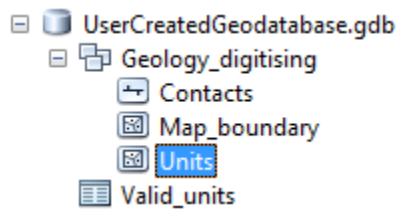
OBJECTID *	Unit_code	Unit_Name
1	Rk	Rakaia Schist
2	SB	Saint Bathans Formation
3	Dn	Dunstan Formation
4	Mto	Maniototo Formation
▶		

7. Save edits and finish editing session

2.2.5 Run Script: "2 Create Geological Unit Polygons"

This script creates geological unit polygons. It performs 3 basic steps: 1) Snapping the **Contacts** to themselves and the **Map_boundary** based on the **Digitising accuracy** (calculated by Field map scale or input by user). 2) Create Unit Polygons for the areas defined by the **Contacts** and **Map_boundary**. 3) Add the Unit Code sub type to the Unit Polygons Feature Class using the **Valid_units** table.





Explanation of input fields

Working Geodatabase

Digitising Feature Dataset (optional)

Contacts Feature Class (optional)

Boundary Feature Class (optional)

Valid Units Table (optional)

Unit Code Field (optional)

Field Map Scale 1:

Digitising Accuracy (Map Units)

Working_Geodatabase: This is the Geodatabase where the template geological features will be created.

Digitising Feature Dataset, Contacts Feature Class, Boundary Feature Class, Valid Units Table, Unit Code field: Optional fields that are the same as what is created by the first script by default.

Field Map Scale 1: Enter the absolute scale of your field map (e.g. 1:1000). This will be used to calculate your digitising accuracy in map units.

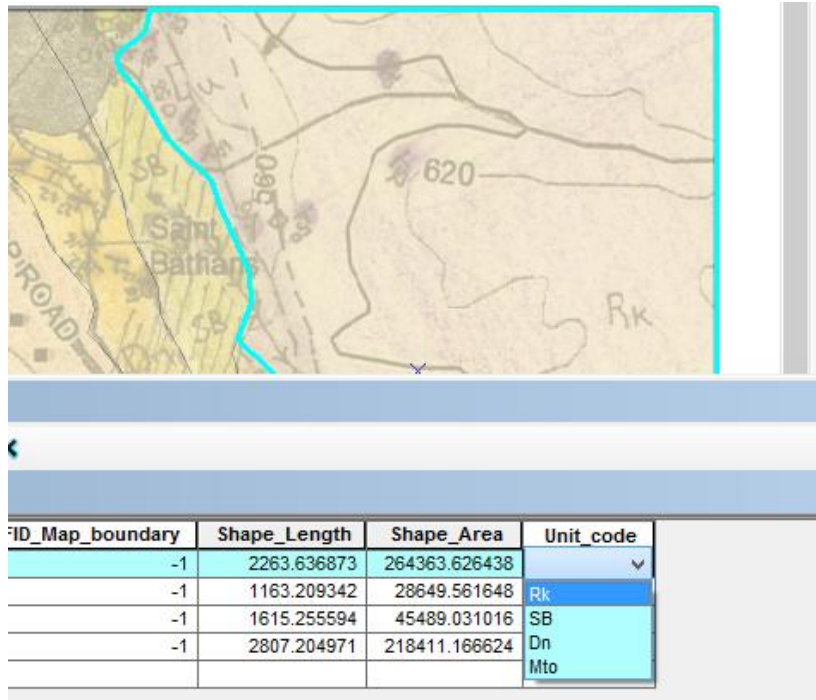
Digitising Accuracy: Calculated from your field map scale by assuming 1/1000 map unit accuracy digitising. e.g. for 1:1000 scale $0.001\text{m} \times 1000 = 1\text{m}$ digitising accuracy. You can override this by setting your map unit accuracy here.

2.2.6 Set unit_code for Geological Units

The last script creates Geological unit polygons that have do not have a Unit Code assigned to them.

1. Add Units to the map
2. Recommended settings for editing Unit_Code:
 - 2.1. Set transparency to 60%
 - 2.2. Make sure base map is visible
3. Start editing session.
4. Open attribute table of units

5. Highlight Row, then choose Correct **Unit_code** based on highlighted polygon on map



ID_Map_boundary	Shape_Length	Shape_Area	Unit_code
-1	2263.636873	264363.626438	▼
-1	1163.209342	28649.561648	Rk
-1	1615.255594	45489.031016	SB
-1	2807.204971	218411.166624	Dn
			Mto

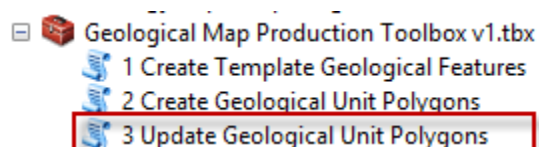
6. Repeat for all remaining polygons.
7. Save edits and end editing session.

2.3 HOW TO USE GEOLOGICAL MAP PRODUCTION TOOLBOX (UPDATE GEOLOGICAL MAP MODE)

There are several ways to update your maps. But the key thing is that you should never edit the Geological Unit Polygons directly apart from setting the geological unit for polygons in the attribute table. This part of the user guide deals with 2 editing scenarios, this script/workflow supports more scenarios than this. In both cases the user performs an editing action and then runs the “**3 Update Geological Unit Polygons**” script.

2.3.1 “3 Update Geological Unit Polygons” Script

This script creates new Geological Unit polygons based on existing Geological polygons with Unit code attribute set, updated contacts and/or Map_boundary and/or Valid_units table. This script is very similar to the “**2 Create Geological Unit Polygons**” script, but has one added step. These are the steps the script performs: 1) Snapping the Contacts to themselves and the **Map_boundary** based on the Digitising accuracy (calculated by Field map scale or input by user). 2) Create Unit Polygons for the areas defined by the **Contacts** and **Map_boundary**. 3) Add the Unit Code sub type to the Unit Polygons using the Valid_units table. 4) Copy the attributes from the existing Geological polygons (Unit Code) to the new Geological Polygons, based on how they overlap. New Unit polygons will have number on end of old polygons incremented by 1. For example if old polygon was Units, new polygon will be Units1.



3 Update Geological

Working Geodatabase
C:\PROCESSING\ArcGIS_processing\Lab3_project\script-test1\

Input Units
C:\PROCESSING\ArcGIS_processing\Lab3_project\script-test1\

Digitising Feature Dataset (optional)
Geology_digitising

Contacts Feature Class (optional)
Geology_digitising/Contacts

Boundary Feature Class (optional)
Geology_digitising/Map_boundary

Valid Units Table (optional)
Valid_units

Unit Code Field (optional)
Unit_code

Field Map Scale 1:
10000

Digitising Accuracy (Map Units)
10

OK Cancel Environments... << Hide Help

Working_Geodatabase: This is the Geodatabase where the template geological features will be created.

Input Units: Find the file with the latest geological unit polygons that you want to update

Digitising Feature Dataset, Contacts Feature Class, Boundary Feature Class, Valid Units Table, Unit Code field: Optional fields that are the same as what is created by the first script by default.

Field Map Scale 1: Enter the absolute scale of your field map (e.g. 1:1000). This will be used to calculate your digitising accuracy in map units.

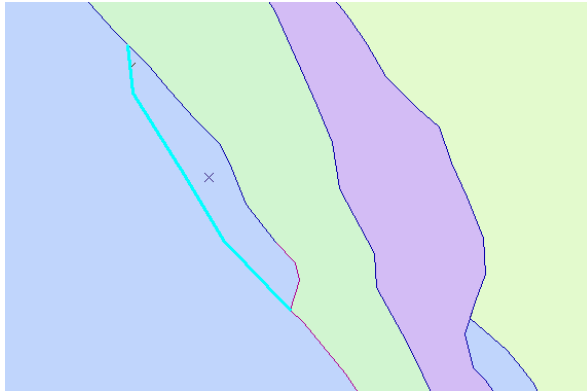
Digitising Accuracy: Calculated from your field map scale by assuming 1/1000 map unit accuracy digitising. e.g. for 1:1000 scale $0.001\text{m} \times 1000 = 1\text{m}$ digitising accuracy. You can override this by setting your map unit accuracy here.

2.3.2 Editing scenario 1: Moving a contact or changing a contact type

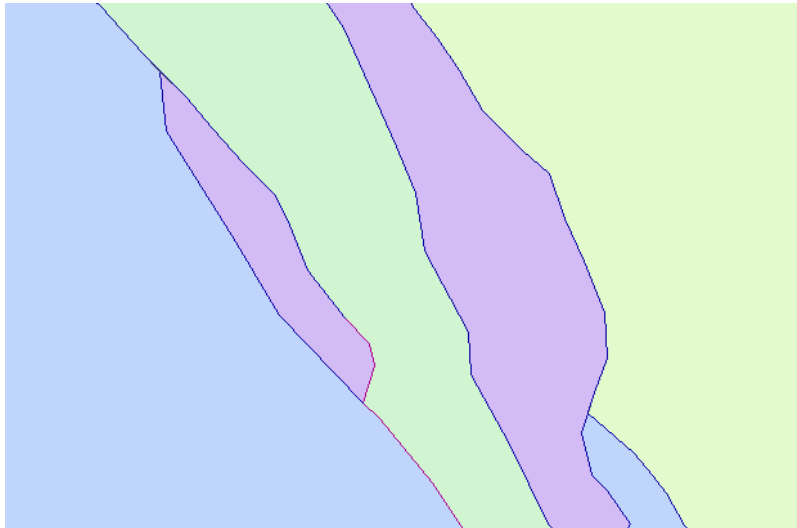
1. Start an edit session (Recommended setting: Snapping Turned on: *End and Edge*)
2. Edit the contact and/or change the type of contact.
3. Save edits and stop editing
4. Run "3 UPDATE GEOLOGICAL UNIT POLYGONS" SCRIPT
5. Check that **Unit_code** is correct for units beside edited area.

2.3.3 Editing scenario 2: Adding a new contact in

6. Start an edit session (Recommended setting: Snapping Turned on: *End and Edge*)
7. Add the new contact/s in



8. Save edits and stop editing
9. Run ""3 UPDATE GEOLOGICAL UNIT POLYGONS" SCRIPT
10. Adding a new contact will create a new Polygon. this will get the attributes of the polygon directly beneath it from the existing Geological Unit Polygons
11. Start an editing session and set what the newly created Geological Unit Polygons should be.



2.4 FUTURE ADDITIONS

These are some features I would like to implement in version 2:

1. Joining the unit description and other information supplied about the units
2. Deal with more geological features for map. Such as point data (foliation, bedding)
3. Turn it into a tool

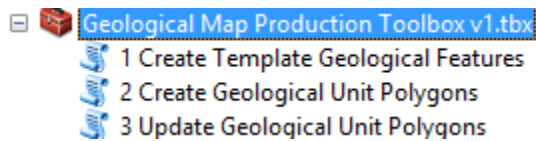
2 PROGRAMMER'S GUIDE

2.5 OVERVIEW

This Arcmap Toolbox consists of three ArcGIS toolbox scripts which connect to 3 python scripts (using a relative path link). Scripts 2 and 3 also have a customised validator which is included as a

separate python file for convenient viewing, but the embedded validator script is used during processing.

- 1 Create Template Geological Features
 - 1_create_template_geological_features_v3.py (python script file)
- 2 Create Geological Unit Polygons
 - 2_create_geological_unit_polygons_v4.py (python script file)
 - 2_create_geological_unit_polygons_v4_validator.py (python script file validator is embedded in ArcGIS script)
- 3 Update Geological Unit Polygons
 - 3_create_output_geology_v2.py
 - 3_create_output_geology_v2_validator.py (python script file validator is embedded in ArcGIS script)



2.6 TOOLBOX FUNCTIONS OVERVIEW

There are two ways to use the Geological Map Production toolbox: 1) In initial production mode, 2) in Update Geological Map mode:

2.6.1 Initial map production mode

Initial map production mode uses “**1 Create Template Geological Features**” and “**2 Create Geological Unit Polygons**”

1 Create Template Geological Features

A feature dataset is created inside the file geodatabase called “**Geology_digitising**” and empty feature classes for contacts and map boundary are created within this feature dataset. An empty table called Valid Units is created with fields for “unit_code” and “Unit_name.”

2 Create Geological Unit Polygons

This script creates geological unit polygons. It performs 3 basic steps: 1) Snapping the **Contacts** to themselves and the **Map_boundary** based on the **Digitising accuracy** (calculated by Field map scale or input by user). 2) Create Unit Polygons for the areas defined by the **Contacts** and **Map_boundary**. 3) Add the Unit Code subtype to the Unit Polygons Feature Class from the **Valid_units** table.

2.6.2 Update Geological Map mode uses

Update Geological Map mode uses “**3 Update Geological Unit Polygons**” script

3 Update Geological Unit Polygons

This script creates new Geological Unit polygons based on existing Geological polygons with Unit code attribute set, updated contacts and/or Map_boundary and/or Valid_units table. This script is very similar to the “**2 Create Geological Unit Polygons**” script, but has one added step. These are the steps the script performs: 1) Snapping the **Contacts** to themselves and the **Map_boundary** based on the Digitising accuracy (calculated by Field map scale or input by user). 2) Create Unit Polygons for the areas defined by the **Contacts** and **Map_boundary**. 3) Add the Unit Code sub type to the Unit Polygons using the **Valid_units** table. 4) Copy the attributes from the existing Geological polygons (Unit Code) to the new Geological Polygons, based on how they overlap. New Unit polygons will

have number on end of old polygons incremented by 1. For example if old polygon was **Units1**, new polygon will be **Units2**.

For a more detailed explanation refer to: **Geological Map Production Toolbox v1 User guide**

2.7 SCRIPTS EXPLAINED

2.7.1 Script style

Most of the scripts used built in **arcpy** functions. Feature datasets, Feature classes and tables are used as input, created or updated in a single working Geodatabase. There are a few loops and one function. The Scripts contain the same error handling code from SURV519 2017 Tutorial 4 instructions. This makes all the code indented. Comments that apply to blocks of code are not indented to make it clear what groups of processing are happening.

```
## Make all the feature datasets and feature classes we will use

#Create the feature dataset where the empty geological features
arcpy.CreateFeatureDataset_management(workingGDBPath, digitising)

#Create the feature class for contacts
```

2.7.2 Code sources

Where code has been used from other sources it is attributed in the script comments. ArcGIS online help is not attributed, but was used extensively for getting the correct syntax for the ArcGIS functions.

2.7.3 Script format

In most cases this format has been used for the code, with slight variations for clarity:

Import modules

Functions

try: #error handling

 Get variables as user input

 Set other variables

 Code for script...

except #error handling

2.7.4 Modules used

```
import arcpy
```

```
from arcpy import env #import for environment setting
```

```
import traceback #for error handling
```

```
import os #import os for easier path operations (e.g. concatenation)
```

```
import numpy #for numpy arrays for Script 2 and 3 only
```

```
import re #for regular expressions in Script 3 only
```

2.7.5 Naming Scheme for variables

ArcPy functions typically use strings (e.g. paths) that point to files rather than generating objects. Or they can use numbers and floats and other types of objects. Keeping track of the names is very important. This script uses this naming scheme for strings that refer to files/fields stored on disc:

“**NameTypeStringtype**”, where the **FileType** may be an abbreviation. For example:

digitisingFDName = digitising[**Name**]FD[*Feature Dataset* **Type**]Name[**Stringtype** is the *Name of a file*]

The **Type** could be a: Geodatabase(GDB), Feature Dataset(FD), Feature Class(FC), Field, Table or other type.

The **Stringtype** could be a “Path” or a “Name” of a file.

2.7.6 Environment setting, path concatenation

All three scripts get the user to select a geodatabase as the **workingGDB**. This enables easy processing and also setting the workspace environment setting. The workspace environment setting essentially allows for the script to call all the files with the path to the **workingGDB** set.

```
##set the environment to the working geodatabase
```

```
env.workspace = workingGDBPath
```

For example in the code below **digitisingFDName** = “**Geology_digitising**” and will be stored in the **workingGDB**. No need to specify where the **workingGDB** is:

```
#Create the feature dataset where the empty geological features  
will be stored
```

```
arcpy.CreateFeatureDataset_management(workingGDBPath,  
digitisingFDName, coordinateSystem)
```

The OS path joining functionality is used to make path concatenation more robust. In the example below the output path will look like this (depending on operating system):

“**digitisingFDName\\contactsFCName**”

```
#Create empty sub type to go in the Accuracy Field
```

```
arcpy.SetSubtypeField_management(os.path.join(digitisingFDName, conta  
ctsFCName), accuracyFieldName)
```

2.7.7 Defaults used in scripts

Both scripts 2 and 3 use defaults for the various input files for processing. These defaults are the same file names created as the templates from Script 1. The user can modify these if they have changed the names of the files and the modified values will be used.

Originally this was meant to be implemented as a checkbox for A: use defaults or B: set your own ones. But the ToolValidator in ArcGIS has some limitations that did not allow for this to work.

2.8 SCRIPT CODE FEATURES

The code is well marked up and fairly simple. It should be easy enough to follow. Here I will highlight a couple of features that may not be too obvious.

2.8.1 Digitising accuracy (script 2 and 3)

The digitising Accuracy field is calculated by the absolute field map scale. Using the **ToolValidator**

The code for the validator is shown below with additional comments. **Field map scale** is `self.params[6].value` **Digitising Accuracy** is `self.params[7].value`

```

if self.params[6].value > 0:

    if not self.params[7].altered:#make sure the user hasn't
altered it. If they have then do not calculate it. This allows the
user to override the calculated value

        self.params[7].value = float(self.params[6].value)*.001
##digitising accuracy is based on 1/1000 accuracy on the map

    return

```

2.8.2 Increment trailing numbers function from script 3.

This block of code looks at a string and uses a regular expression to get the trailing numbers off the end. e.g. "sam54" turns into "54". If there is a number it gets incremented by 1, if there is not a number the number 1 gets added to the end. The purpose of this code is to allow the user to keep

regenerating the Units polygons and be able to compare to the last one they used as input (i.e. not overwriting it).

```
##code to automatically increment trailing numbers on a string.

##Modified from https://stackoverflow.com/questions/7085512/check-
what-number-a-string-ends-with-in-python

##Luke added functionality to make it increment the number, rather
than just fetch it

import re

def increment_trailing_number(s):

    m = re.search(r'\d+$', s)#regular expression to find the numbers
on the end

    newString = ""

    if m:

        n = int(m.group())

        newString = s[:-1*len(str(n))] + str(n+1)

    else:

        newString = s + str(1)

    return newStr
```