Modifying DOCX custom document properties

Gerald Q. Maguire Jr.

Given that the thesis template and the integrated cover and thesis template have a set of custom Doc Properties (here after DocProp), one thought was to be able to generate pre-customized DOCX files that could contain information that was already known. For example, if one knows the name of the student or students who will be the authors this information could be "injected" into the custom DocProp file. The idea was that such a pre-configured file could be given to the student and they would not need to enter this information themselves.

1 Extracting custom DocProp

Extracting custom DocProp from an existing file can be done using a program such as `extract_custom_DOCX_properties.py`, as shown in Figure 1. In this case zb1.docx is a copy of `Template-thesis-English-2022-with-cover_and_for-DiVA-v2.docx` (the integrated cover and thesis template. This produced a JSON file whose contents are shown in Figure 2.

```
./extract_custom_DOCX_properties.py --file zb1.docx
```

Figure 1: Command to extract the existing custom DocProp from a DOCX file

```
{"2": {"ZOTERO_PREF_1": "&lt;data data-version=\"3\" zotero-version=\"4.0.29.16\"&gt;&lt;session
id=\"txwJq0tU\"/&gt;&lt;style id=\"http://people.kth.se/~maguire/ExampleStyle-with-access.csl\"
hasBibliography=\"1\" bibliographyStyleHasBeenSet=\"1\"/&gt;&lt;prefs&gt;&lt;pref name=\"fieldType\"
value=\"Field\"/&gt;&lt;pref name=\""}, "3": {"ZOTERO_PREF_2": "storeReferences\"
value=\"true\"/&gt;&lt;pref name=\"automaticJournalAbbreviations\" value=\"true\"/&gt;&lt;pref
name=\"noteType\" value=\"\"/&gt;&lt;/prefs&gt;&lt;/data&gt;"}, "4": {"Subtitle": "An subtitle in the
language of the thesis"}, "5": {"Alternative_main_title": "Detta är den svenska översättningen av
titeln"}, "6": {"Alternative_subtitle": "Detta är den svenska översättningen av undertiteln"}, "7":
{"Author1_Last_name": "Student"}, "8": {"Author1_First_name": "Fake A."}, "9": {"Author1_Local User
Id": "u100001"}, "10": {"Author1_E-mail": "a@kth.se"}, "11": {"Author1_organization_L1": "School of
Electrical Engineering and Computer Science"}, "12": {"Author1_organization_L2": "&lt;NA&gt;"}, "13":
{"Author1_Other_organisation": "&lt;NA&gt;"}, "14": {"Author2_Last_name": "Student"}, "15":
{"Author2_First_name": "Fake B."}, "16": {"Author2_E-mail": "b@kth.se"}, "17":
{"Author2_organization_L1": "School of Architecture and the Built Environment"}, "18":
{"Author2_organization_L2": "&lt;NA&gt;"}, "19": {"Author2_Other_organisation": "&lt;NA&gt;"}, "20":
{"Author2_Local User Id": "u100002"}, "21": {"Cooperation_Partner_name": "Företaget AB"}, "22":
{"Cycle": "1"}, "23": {"Course_code": "IA150X"}, "24": {"Credits": "15"}, "25": {"programcode":
"TCOMK"}, "26": {"Educational program": "Bachelor's Programme in Information and Communication
Technology"}, "27": {"Degree": "Bachelors degree"}, "28": {"subjectArea": "Information and
Communication Technology"}, "29": {"Second_programcode": "LÄGER"}, "30": {"Second_Educational_program":
"Second_Educational_program"}, "31": {"Second_degree": "Master of Science in Engineering and in
Education"}, "32": {"Second_subjectarea": "Mathematics and Chemistry"}, "33": {"Examiner1_Last_name":
"Maguire Jr."}, "34": {"Examiner1_First_name": "Gerald Q."}, "35": {"Examiner1_Local User Id":
"u1d13i2c"}, "36": {"Examiner1_E-mail": "maguire@kth.se"}, "37": {"Examiner1_organization_L1": "School
of Electrical Engineering and Computer Science"}, "38": {"Examiner1_organization_L2": "Computer
Science"}, "39": {"Supervisor1_Last_name": "Supervisor"}, "40": {"Supervisor1_First_name": "A. Busy"},
"41": {"Supervisor1_Local User Id": "u100003"}, "42": {"Supervisor1_E-mail": "sa@kth.se"}, "43":
{"Supervisor1_organization_L1": "School of Electrical Engineering and Computer Science"}, "44":
{"Supervisor1_organization_L2": "Computer Science"}, "45": {"Supervisor1_Other_organisation":
"&lt;NA&gt;"}, "46": {"Supervisor2_Last_name": "Supervisor"}, "47": {"Supervisor2_First_name": "Another
Busy"}, "48": {"Supervisor2_Local User Id": "u100003"}, "49": {"Supervisor2_E-mail": "sb@kth.se"},
"50": {"Supervisor2_organization_L1": "School of Architecture and the Built Environment"}, "51":
{"Supervisor2_organization_L2": "Architecture"}, "52": {"Supervisor2_Other_organisation":
"&lt;NA&gt;"}, "53": {"Supervisor3_Last_name": "Supervisor"}, "54": {"Supervisor3_First_name": "Third
Busy"}, "55": {"Supervisor3_Local User Id": "uxxxxx"}, "56": {"Supervisor3_E-mail": "sc@tu.va"}, "57":
{"Supervisor3_organization_L1": "&lt;NA&gt;"}, "58": {"Supervisor3_organization_L2": "&lt;NA&gt;"},
"59": {"Supervisor3_Other_organisation": "Timbuktu University, Department of Pseudoscience"}, "60":
{"Opponents_Name": "A. B. Normal &amp;amp; A. X. E. Normalè"}, "61": {"Presentation_Date": "2021-03-15
13:00"}, "62": {"Presentation_Language": "eng"}, "63": {"Presentation_Room": "via Zoom https://kth-
se.zoom.us/j/ddddddddddd"}, "64": {"Presentation_Address": "Isafjordsgatan 22 (Kistagången 16)"}, "65":
{"Presentation_City": "Stockholm"}, "66": {"Series_name": "EECS-EX"}, "67": {"Number_in_series":
"2021:00"}, "68": {"National Subject Categories": "10201, 10206"}}
```


Figure 2: `zb1-extracted.json`

The extraction works by getting the names of the DocProp and their values from the file within the ZIP archive of the DOCX file. The specific file is: '`docProps/custom.xml`'.

2 Generate a modified JSON file with the DOCProp contents that you want

Next we generated a JSON file that will be used later to modify a DOCX file. In this case, the file is called `zb1-customize.json`. In this case, the first names of the two authors have been changed. These changes are highlighted in the figure.


```
{"7": {"Author1_Last_name": "Student"}, "8": {"Author1_First_name":
"Fake C."}, "9": {"Author1_Local User Id": "u100001"}, "10":
{"Author1_E-mail": "a@kth.se"}, "11": {"Author1_organization_L1":
"School of Electrical Engineering and Computer Science"}, "12":
{"Author1_organization_L2": "&lt;NA&gt;"}, "13":
{"Author1_Other_organisation": "&lt;NA&gt;"}, "14":
{"Author2_Last_name": "Student"}, "15": {"Author2_First_name": "Fake
D."}, "16": {"Author2_E-mail": "b@kth.se"}}
```


Figure 3: `zb1-customize.json` (showing just part of the file).

3 Modifying the DOCX file

The next step is to actually make the desired changes in the DocProp values. This is done using the program `customize_DOCX_file.py` with a command such as shown in Figure 4. The command produces a file called `zb1-modfied.docx`.

```
./customize_DOCX_file.py --json zb1-customize.json --file zb1.docx
```

Figure 4: Command to customize a DOCX file

4 Viewing the modified DOCX file

When we view the file zb1-modfied.docx in Word we see the unexpected results shown in Figure 5. This is unexpected because the first names of the authors do not appear to be changed.

Unfortunately, even if we do **`Cntrl-A`** (to selected all the content in the document) and then type **`F9`** to update all of the fields, the fields are **not** updated! However, if we explicitly selected the contents of the authors control box and type **`F9`**, the fields are updated.

So what is going on? It seems that this is a problem with Word and is well document in many postings about this problem. There are three solutions: (1) find all of the fields and manually selected them and update the fields, (2) use a macro (such as shown in Figure 6 or Figure 7)*, or (3) simply set up the document to update fields before printing (see Figure 8 and Figure 9) and then initiate printing. The results is shown in

Degree·project·in·Information·and·Communication·Technology¶

First·cycle,·15·HP ¶

This·is·the·title·in·the·language·of·the·thesis¶

An·subtitle·in·the·language·of·the·thesis¶

FAKE·A.·STUDENT·AND·FAKE·B.·STUDENT¶

Figure 5: Initial view of the cover page

```
Sub UpdateAllFields()
    With ActiveDocument
        .PrintPreview
        .ClosePrintPreview
    End With
End Sub
```

Figure 6: Macro to update all of the fields in a document (from https://wordribbon.tips.net/T013475_Updating_Fields_Automatically.html)

* An even more elegant was is shown in https://wordribbon.tips.net/T013475_Updating_Fields_Automatically.html

```
Sub UpdateAllFields()
Application.ScreenUpdating = False
With ActiveDocument
  .Fields.Update
  .PrintPreview
  .ClosePrintPreview
End With
Application.ScreenUpdating = True
End Sub
```

Figure 7: An alternative macro document (from https://answers.microsoft.com/en-us/msoffice/forum/all/updating-all-fields-in-a-word-document/4462e35a-6284-4804-bd38-0514c852b616)

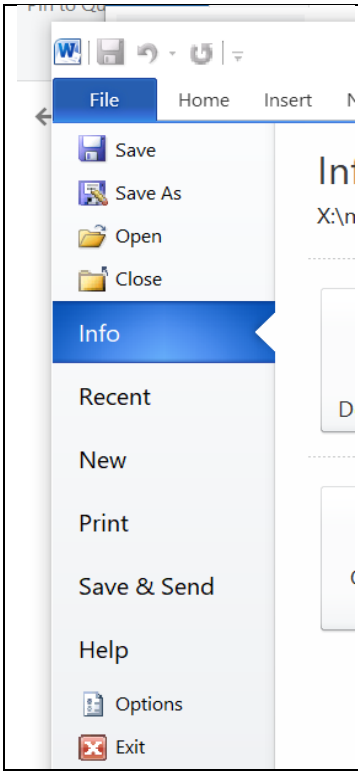

Figure 8: Setting the options via the File menu

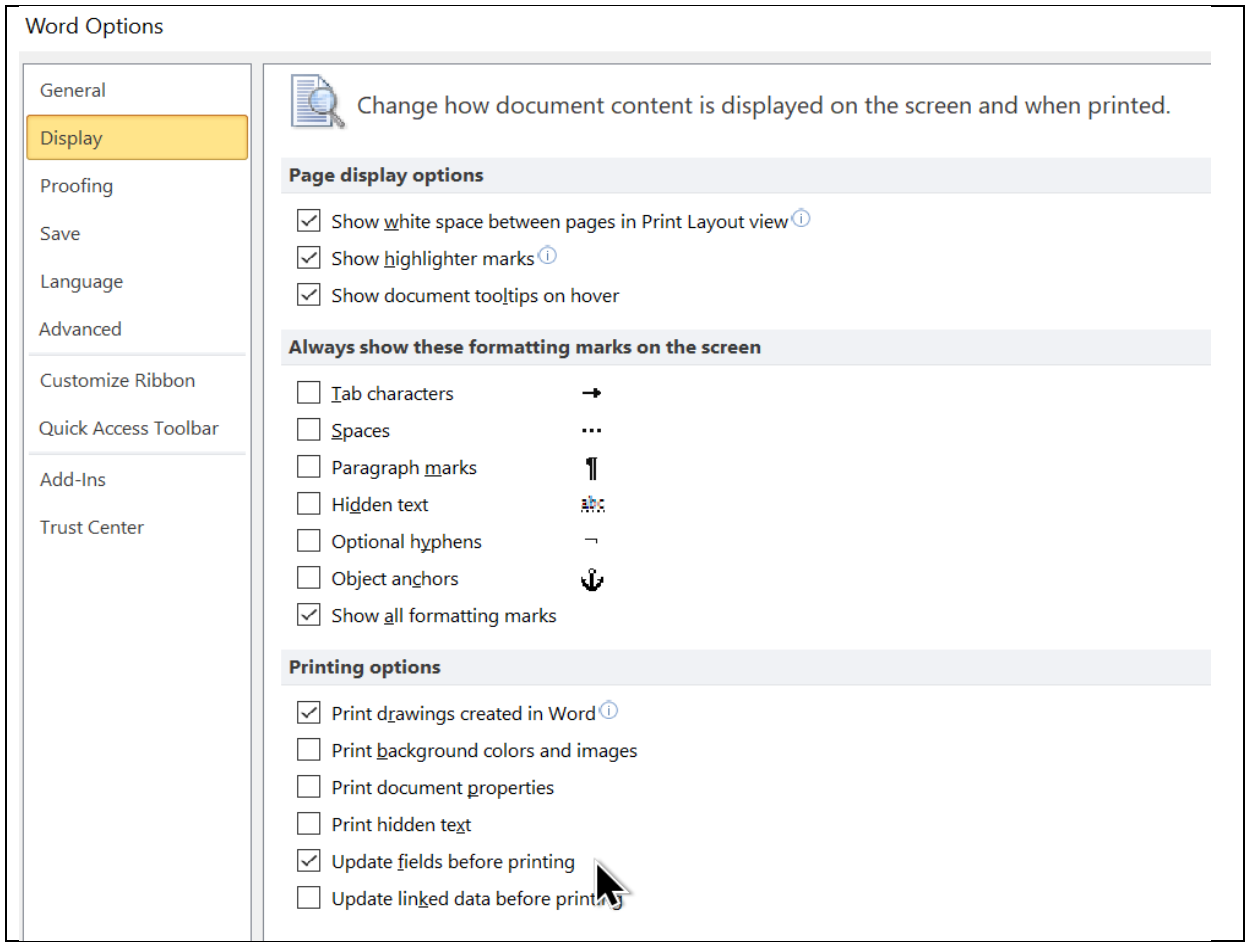

Figure 9: Setting the options to update fields before printing

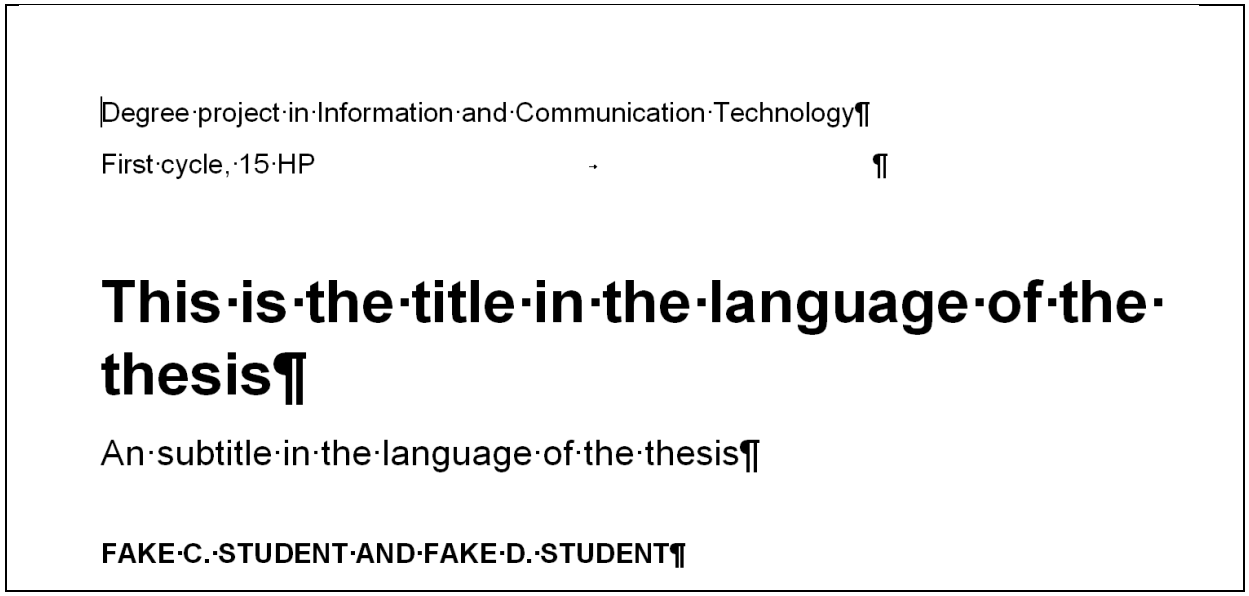

Figure 10: Resulting cover with the desired changes

Note that the same problem affects the title page as can be seen in the *before* printing picture in Figure 11 and the results *after* printing (Figure 12).

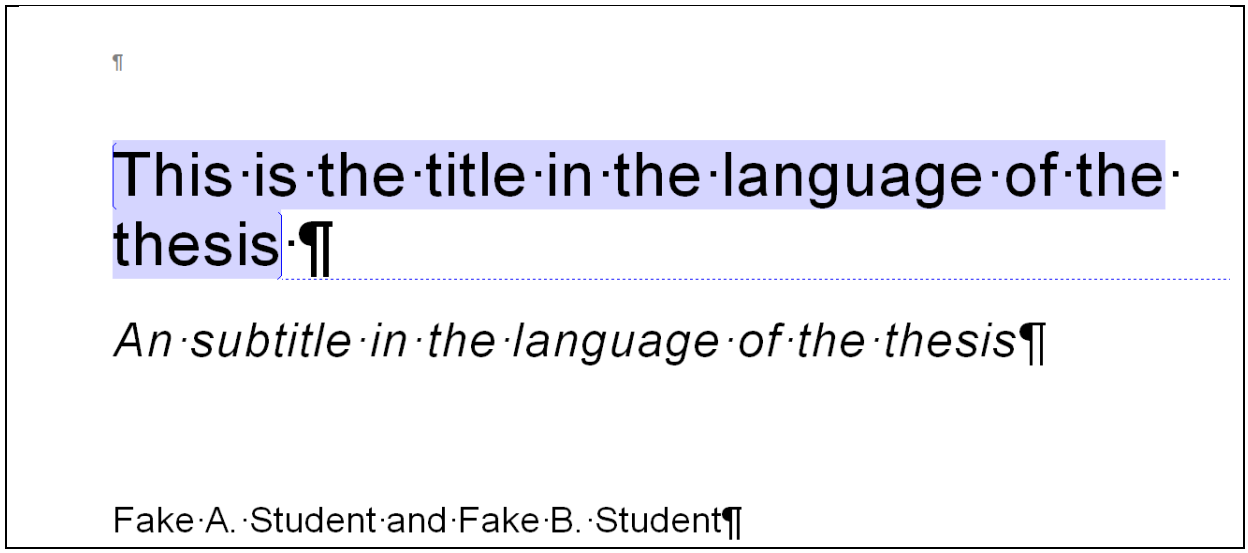

Figure 11: Inside title page after making the changes but before printing

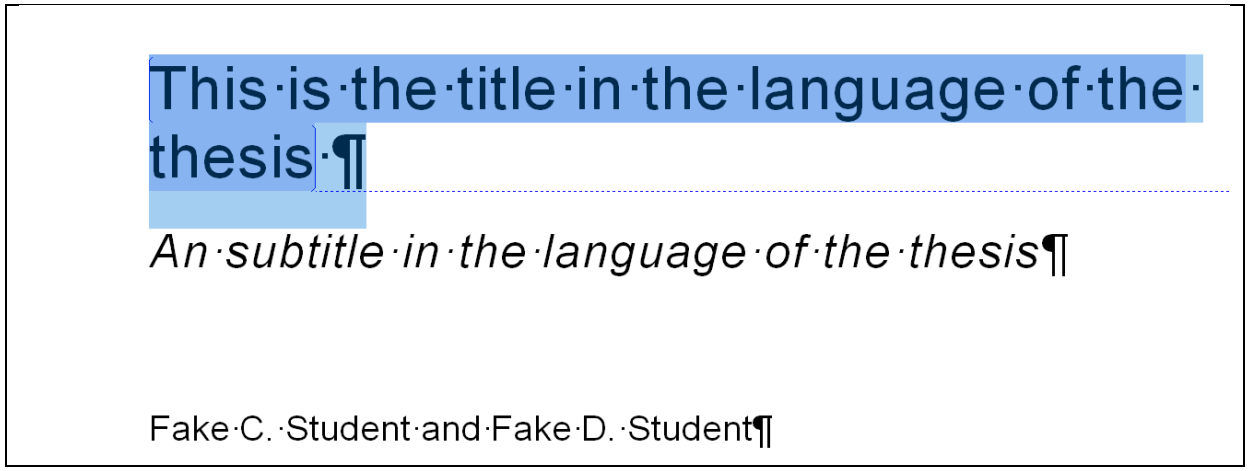

Figure 12: Inside title page after making the desired changes and triggering printing

5 Conclusions to this point

While the feasibility of pre-configuring a DOCX file has been shown, it is less than smooth – since the user has to actively print the file to easily see the modified document. One possible way of hiding this complexity is to create the preconfigured file, programmatically do a print, and then use the resulting updated DOCX file as the preconfigured file to be used by the student.

While the examples in this document have used an integrated thesis with cover and thesis template, this approach of modifying custom DocProp could be applied to any document where there are custom DocProp values. Some examples, of alternative applications are customized forms, customized exams, customized written assignments.

6 What can be done?

Well it turns out that the existing data for the field codes is in the document in a complex field code. An example from my sample document is shown in Figure 13.

```
<w:r>
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r w:rsidRPr="00667DA4">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
    <w:lang w:val="en-US"/>
  </w:rPr>
  <w:instrText xml:space="preserve"> DOCP
  </w:instrText>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
    <w:lang w:val="en-US"/>
  </w:rPr>
  <w:instrText>ROPERTY  Author1_First_name
  </w:instrText>
</w:r>
<w:r>
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:proofErr w:type="gramStart"/>
<w:r w:rsidRPr="00667DA4">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
    <w:lang w:val="en-US"/>
  </w:rPr>
  <w:t>Fake A.
  </w:t>
</w:r>
<w:proofErr w:type="gramEnd"/>
<w:r>
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

Figure 13: Example of a complex field code

Another example from "Wordprocessing Fields" by Daniel Dick at his Office Open XML site (http://officeopenxml.com/) is a complex field to enter a date. The XML is shown in Figure 14. He cites as a reference: ECMA-376, 3rd Edition (June, 2011), Fundamentals and Markup Language Reference § 17.16.18. He points out that w:fldChar has three attributes: dirty, fldCharType, and fldLock. If fldLock is specified as true then the field should not be updated, otherwise (including if it is absent) fldLock is assumed to be false. The standard described three values for fldCharType: start, end, and separate. The first two indicate the start and end of the field while separate specifies a separator character to end the field code or function and start the value (in this case the run containing the text of the date. This leaves the very interesting attribute dirty. The dirty attribute specified where the field needs to be updated. This is always placed in the <w:fldChar> element where the attribute w:fldCharType="begin"/ is placed.

```
<w:r>
<w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r>
<w:instrText xml:space="preserve"> DATE </w:instrText>
</w:r>
<w:r>
<w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r>
<w:t>MM/DD/YYYY</w:t>
</w:r>
<w:r>
<w:fldChar w:fldCharType="end"/>
</w:r>
```

Figure 14: Example of a complex field code for DATE inspired by http://officeopenxml.com/WPfields.php

An example of a complex checkbox field checkbox field is shown in WordprocessingML at https://docs.microsoft.com/en-us/dotnet/api/documentformat.openxml.wordprocessing.fieldcode?view=openxml-2.8.1.

The <w.instrText> element (defined in ECMA-376, 3rd Edition (June, 2011), Fundamentals and Markup Language Reference § 17.16.23) contains the field codes. Daniel Dick describes these field codes at http://officeopenxml.com/WPfieldInstructions.php. The DOCPROPERY shown in Figure 13 falls into the document information category of field codes. Note that you may have to combine multiple separate <w.instrText> elements to get the complete field code, as shown in Figure 15.

```
<w:r w:rsidRPr="00667DA4">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
    <w:lang w:val="en-US"/>
  </w:rPr>
  <w:instrText xml:space="preserve"> DOCPROPERTY  Author1_First_name
  </w:instrText>
</w:r>
```

Figure 15: Complete <w.instrText> (combing the separate runs with the two <w.instrText> elements)

The proofErr elements, with the attributes `w:type="gramStart"/` and `w:type="gramEnd"/` delimit the start and end of the content that indicates a grammatical error. In the case shown in Figure 13 (and reproduced in Figure 16), this is meaningless as it is the text for a person's name; hence, it

should not be grammar checked at all. However, it is the value within the `<w:t>` within the bounded run that we want to update.

```
<w:proofErr w:type="gramStart"/>
<w:r w:rsidRPr="00667DA4">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
    <w:lang w:val="en-US"/>
  </w:rPr>
  <w:t>Fake A.
  </w:t>
</w:r>
<w:proofErr w:type="gramEnd"/>
<w:r>
```

Figure 16: Example of <w:prooferr> delimiting a run containing a <w:t> element with the text of the value of the DOCPROPERTY Author1_First_name

This suggests that one can edit these `<w:t>` values and when the report is opened it will show the correct string. This works as shown in Figure 17and Figure 18. Note that because of the `IF` statement involving the second author's lastname (i.e., `DOCPROPERTY Author2_Last_name`) the situation is more complex that one might think. For details, see Figure 19.

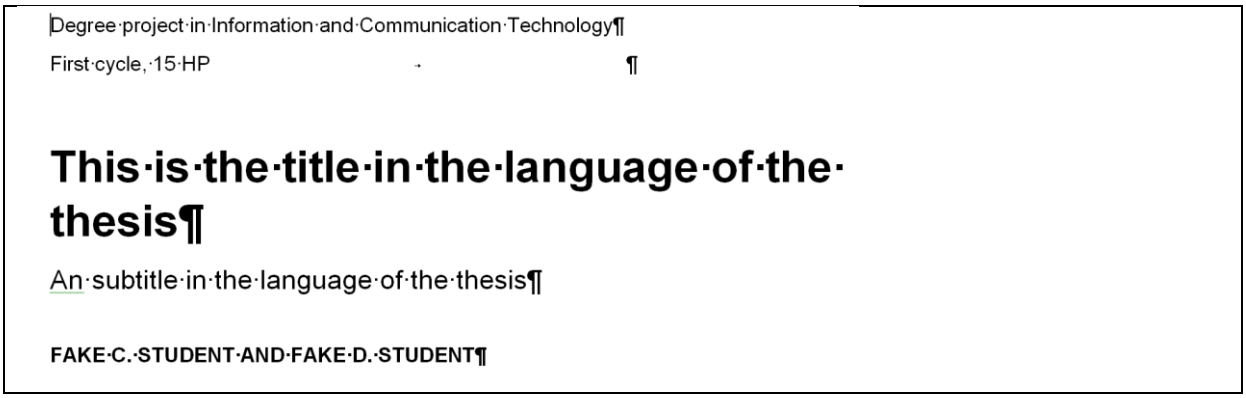

Figure 17: Cover after manual edit of the `<w:t>` elements

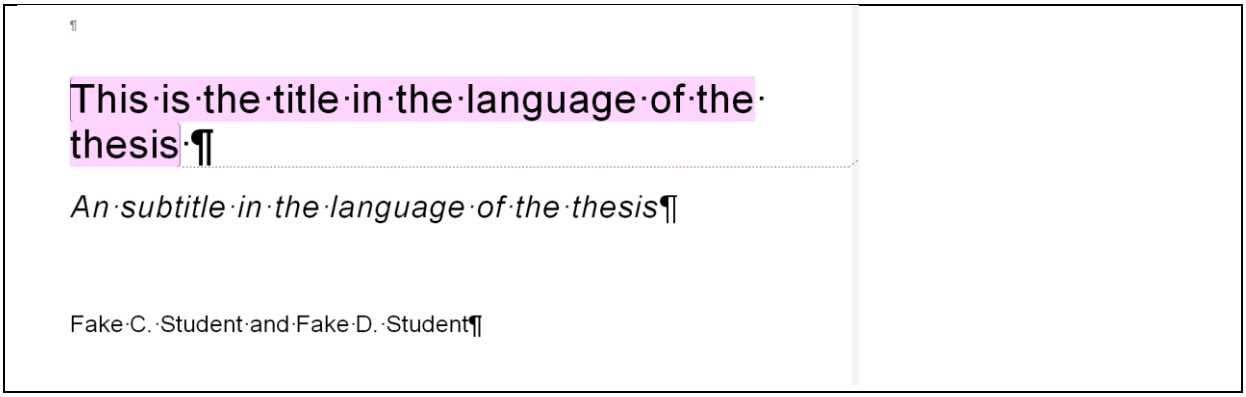

Figure 18: Title page after manual edit of the <w:t> elements

```
<w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:instrText xml:space="preserve"> IF "
  </w:instrText>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:instrText xml:space="preserve"> DOCPROPERTY  Author2_Last_name
  </w:instrText>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:instrText>Student
  </w:instrText>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:fldChar w:fldCharType="end"/>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:instrText xml:space="preserve">" = "" "" "and
  </w:instrText>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:instrText xml:space="preserve"> DOCPROPERTY  Author2_First_name
  </w:instrText>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:fldChar w:fldCharType="separate"/>
```

```
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:instrText>Fake D.
  </w:instrText>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:fldChar w:fldCharType="end"/>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:instrText xml:space="preserve">
  </w:instrText>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:instrText xml:space="preserve"> DOCPROPERTY  Author2_Last_name
  </w:instrText>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:instrText>Student
  </w:instrText>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:fldChar w:fldCharType="end"/>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:instrText>"
  </w:instrText>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
```

```
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
    <w:noProof/>
  </w:rPr>
  <w:t>and Fake D. Student
  </w:t>
</w:r>
<w:r w:rsidR="00784A17">
  <w:rPr>
    <w:rFonts w:cs="Arial"/>
  </w:rPr>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

Figure 19: Example of a complex field code with a conditional IF

However, it turns out that if you set the first <w:fldChar> to be **dirty** (as shown in Figure 20), then when you open the document it will prompt you to update – as shown in Figure 21 to Figure 24.

```
<w:fldChar w:fldCharType="begin" w:dirty="true"/>
```

Figure 20: Setting a <w:fldChar> to be dirty

Figure 21: First prompt for updating when opening the file

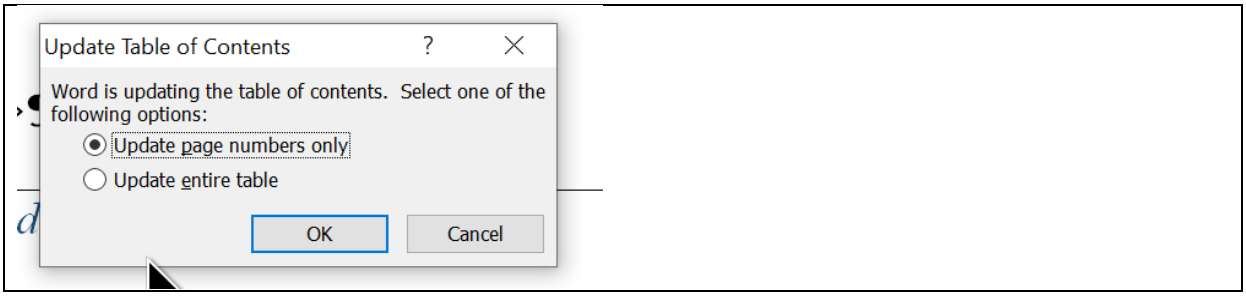

Figure 22: A prompt about updating the table of contents

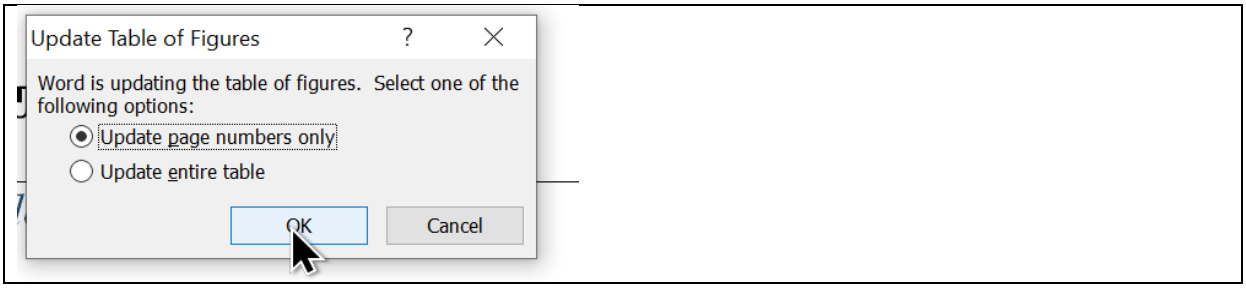

Figure 23: Another prompt to update the table of figures

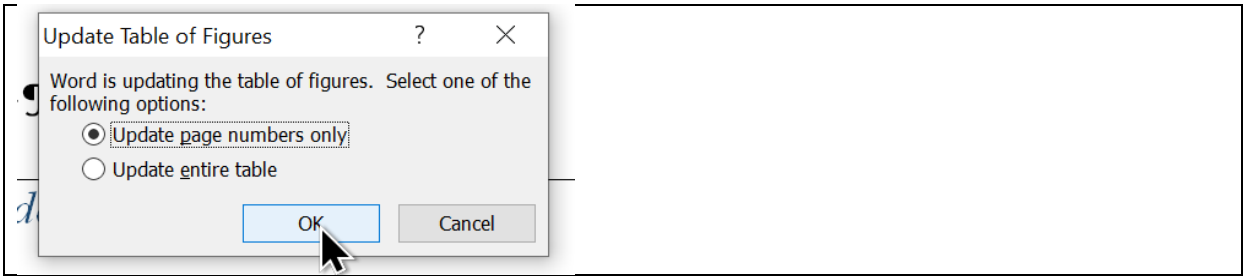

*Figure 24: **Another** prompt about updating the table of figures*

It is unclear to my why you get a second prompt about updating the figures (as was shown in Figure 24). However, if you say Yes and OK to each of the prompts about updating, the resulting updated cover and title page are shown in Figure 25 and Figure 26.

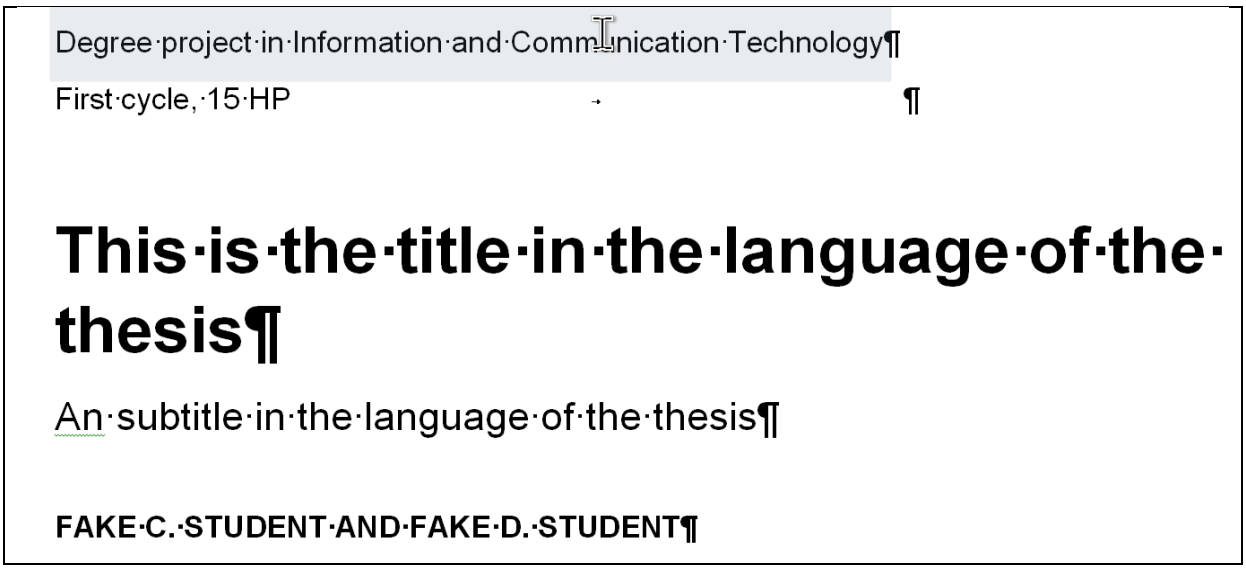

Figure 25: Updated cover page after setting <w:fldChar> to be dirty

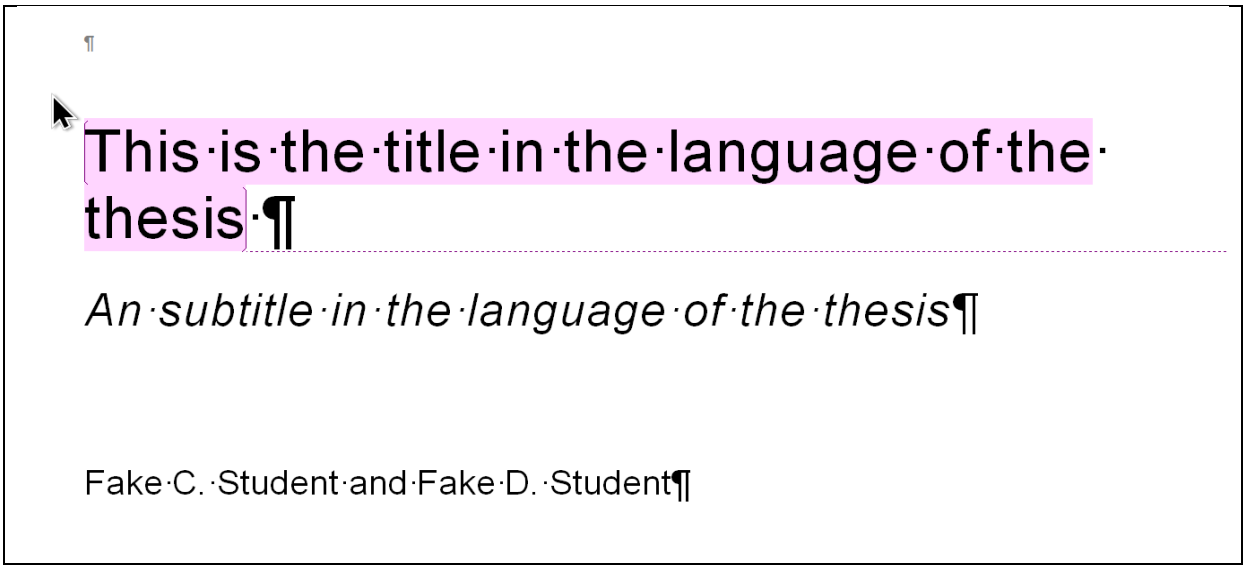

Figure 26: Updated title page after setting <w:fldChar> to be dirty

The interesting part about this was no need to actually go about updating the cached version of the fields, I **only** had to indicate that one of the fields was dirty and this caused all of the fields to be updated. This makes the process really easy.

7 Conclusion

Modifying the DocProp to change the values of fields is not only feasible, but actually turns out to be easy and one does not even need to fake or do printing!