カスタムＸＭＬを用いた Word VBA 不正行為防止マクロの開発

著者名(日)	太田 康友
著者名よみ	オオタ ヤストモ
雑誌名	駿河台大学論叢
号	64
ページ	41-55
発行年	2023
URL	http://doi.org/10.15004/00002499

カスタム XML を用いた Word VBA 不正行為防止マクロの開発

太　田　康　友

I.　はじめに

1.　本研究の背景

Covid-19 の感染拡大によって、筆者が担当する駿河台大学の情報基礎科目は 2020 年度および 2021 年度をフルオンデマンド型授業の形態で実施した。一定レベルのスキル定着を到達目標とする情報基礎科目においては、通常のレポート課題とは異なり、お手本通りのファイル作成を要求する課題が重要であるが、完成度の高い提出物であれば誰が作成した課題ファイルであってもほぼ同じものが出来上がる性質があるため、他人の作成したファイルを提出する安易な不正行為を防止するための対策が急務となっていた。

オンデマンド型授業における不正行為防止対策としては、例えば関西学院大学のようにスマートフォンと連携した AI による顔認証システムを導入することも考えられるが[1]、そういったシステムの導入は全学的な取組みであること、コストが掛かることなどから、教員個人レベルでの導入は困難であることが課題であった。

2.　本研究の目的

本研究の目的は Word ファイル形式での提出課題における不正行為を防止することである。不正行為には様々な形態が考えられるが、その中でも頻度の高い「他人の作成したファイルを提出する」「他人の作成したファイルからコピー＆ペーストする」といった安易な不正行為を防止することを主眼に置いた。また、「教員個人レベルで」「安価に」「不正行為を防止できる」仕組みとして Visual Basic for Applications（以下 VBA）を用いた Word マクロを選択し開発を行った。

II.　不正行為防止マクロの概要

本研究で開発した不正行為防止マクロは、Windows、Mac に関わらず動作することが必要であったため VBA の機能のみを用いることとし、OS 固有の API は一切呼び出していない。以下に Word 不正行為防止マクロの概要を示す。

1.　カスタム XML を利用した作業ログの記録

Excel ファイル形式においては、学生に入力させた学籍番号を非表示シート（学生からは見えない場所）に記録する方式が提案されているが[2]、Word ファイル形式に同様の機能はない。そこで、Office2007 以降のファイル形式で扱うことができるカスタム XML の機能に着目し、作業ログをカスタム XML に記録・保存するよう実装した。

カスタム XML は Word ファイルの内部に作成される XML 形式のファイルである。Word ファイルの拡張子を .zip に変更して展開すると内部構造が見える。図 1 の「customXml」がカスタム XML のファイルが保存される内部フォルダであり、通常の Word ファイルには存在しないが VBA によってカスタム XML を作成するとフォルダが作られる。このカスタム XML ファイルに、学生に入力させた「学籍番号」「氏名」と PC 環境に関するデータである「Office ユーザー名」、そして不正行為判定のためのメタデータとして「作業開始時刻」「保存時刻」を作業ログとして記録する。

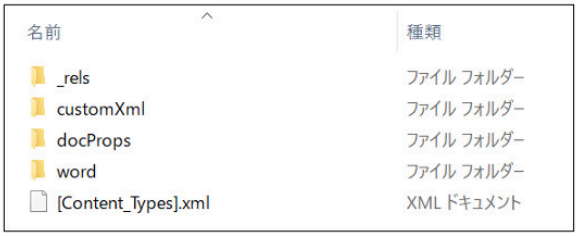

図 1. Word ファイルの内部構造：customXML フォルダは通常の Word ファイルには存在しないが、VBA によって作成される

2.　動作機序

学生は配布した Word マクロファイルを用いて課題に取り組ませる。配布ファイルはパスワードによる編集制限を掛けた状態で配布することで、学生はマクロを有効化し、学籍番号と氏名を入力しなければ編集できないため、必ず何らかの入力が必要となる。この動作機序によって、他人が作成したファイルをそのまま提出された場合、作業ログを確認することで不正行為を容易に判断できるようになっている。

その他、簡易的なコピー ＆ ペースト防止機能も実装している。他のアプリケーションウィンドウから本マクロを実装した Word ファイルのウィンドウにフォーカスが移ったタイミングで Word 本文中から 1 文字コピーすることで、クリップボードを上書きする。非常に簡易的な機能であるため回避も可能であるが、安易なコピー ＆ ペーストは十分に防ぐことができる。

III.　Word 不正行為防止マクロの使い方

1.　Word 不正行為防止マクロ（学生配布用）の準備

Wordマクロ有効文書(.docm形式)を作成する。Visual Basic Editor（以下 VBE）を起動し、標準モジュール以下 Modle1 に「別添資料 1：Module1 ソースコード」を貼り付けて行番号を削除する。VBE メニューの【挿入】>【クラスモジュール】でクラスモジュールを追加した上で、クラスモジュール以下 Class1 に「別添資料 2：Class1 ソースコード」を貼り付けて行番号を削除する。この際、YOURPASSWORD と書かれた箇所はすべて、オリジナルの同一パスワードに変更しておく。最後に念のため、VBE メニューの【ツール】>【Project のプロパティ】から、ソースコードの表示にパスワードロックを掛けておくことが望ましい。

また、配布の際にはマクロを無効化した状態では編集できないようにするため、Word の【校閲】タブ>【編集の制限】から、マクロ内に記述したものと同一のパスワードで編集の制限を掛けておく必要がある。

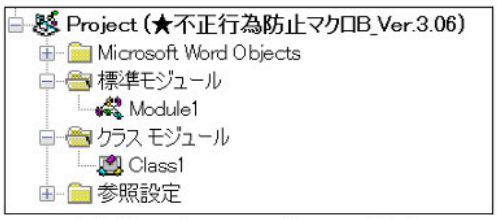

図 2. マクロのモジュール構成

2.　作業ログ確認用マクロ（教員用）の準備

学生の作業ログを確認するためのマクロ「別添 3 normal.dotm 確認用マクロ」を、Word の Normal.dotm 内の Module1 に登録する。Normal.dotm は Word の標準テンプレートであり、使用する PC ごとに登録する必要がある。YOURPASSWORD と書かれた箇所は Word 不正行為防止マクロ（学生配布用）と同一のものに変更しておく。

Normal.dotm へのマクロ追加方法は、VBE において Normal を選択した状態で【挿入】>【標準モジュール】とした後、Normal 以下の Module1 に別添資料 3 のソースコードを貼り付ければ良い。

3.　学生へのファイル配布と回収後のフロー

(1)　学生への指示

手順通りに作成した配布用ファイルは、学生が配布ファイルのマクロを有効にした上で学籍番号および氏名を入力しなければ編集できない状態となっている。学生がマクロを有効にするための準備として、Word の【ファイル】>【オプション】

>【トラストセンター】>【マクロの設定】において「警告を表示してすべてのマクロを無効にする」をあらかじめ設定させる必要がある。また、配布用ファイルを開いた際には【コンテンツの有効化】をクリックしてマクロを有効化することを指示する。

本原稿の執筆時点では、インターネットから取得したファイルについて VBA 機能がデフォルトで無効化されるようになったため、配布ファイルのプロパティで【セキュリティ：】を許可する必要もある。

これらの手順については学生向けに別途マニュアルを用意し配布するべきである。

(2)　回収ファイルのチェック

学生から提出されたファイルの確認には Normal.dotm に登録した「カスタム XML チェック」マクロを起動する。ポップアップ表示されるウィンドウには、上から順に「入力された学籍番号」「入力された氏名」「Office ユーザー名」「ファイルを開いたときの日時」「保存時の日時」が表示される。（図 4）

なお、「カスタム XML チェック」マクロを Word のクリックアクセスツールバーの一番左に追加しておくと Alt + 1 のショートカットキーで呼び出すことができるため、チェック作業を手軽に行うことができる。

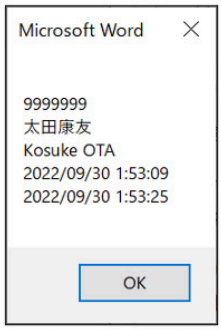

図 3. カスタム XML チェック画面の表示例

IV.　実装の解説

ソースコードを改変しやすいよう、実装について簡単に解説する。ソースコードはそれぞれ別添資料を参照のこと。

1.　Sub AutoOpen（Module1：行 16-118）

起動時に自動実行されるプロシージャであり、学籍番号と氏名を入力させることがメイン処理である。制御用フラグ変数の設定をした後、「マクロを有効にして学籍番号と氏名を入力しない限り、ファイルを編集できない」状態を担保するために編集を制限して上書き保存している（行 40-54）。

その他に、学生がマクロ有効な状態を視覚的に確認できるよう Word 文書のヘッダー部に学籍番号と氏名を表示する処理をしており、エラー回避のために印刷レイアウトへの変更等の前処理を含む一連の流れとなっている。ややわかりにくい順序で処理されている理由は印刷レイアウトへの変更は時間が掛かる処理であるためである。また、OneDrive 上では自動保存がデフォルトで有効となっており、自動保存においては保存日時が記録されないことから、行 32-35 で自動保存を解除している。

行 61-72 の学籍番号取得については、マクロ利用の際に注意が必要である。駿河台大学においては学部生の学籍番号は数字 7 桁であることから半角数字 7 桁以外の入力を受け付けない処理となっている。学籍番号にアルファベットを含む教育機関で利用する場合は、行 63-72 をコメントアウトするなど適宜オミットが必要である。

2.　Sub AtOpen（Module1：行 139-158）

ファイルを開いた直後、カスタム XML に学籍番号、氏名、開始日時を記録するプロシージャであり、AutoOpen から呼び出される。

3.　Sub AtSave（Module1：行 161-179）

ファイル保存イベント捕捉時、Word アプリケーションが保存を実行する直前に Class1 モジュールから呼び出されるプロシージャであり、カスタム XML に保存時刻を追記する。また、Word アプリケーションの強制終了があった場合においてもファイルが編集制限された状態となるよう、保存処理の前に編集制限を掛けている（行

175-178）。

4.　Sub AfterSave（Module1；行 182-204）

ファイル保存イベント終了 1 秒後に呼び出されるプロシージャであるり、学生がファイル編集できるよう編集制限を解除している。

5.　Sub CopyWord（Module1：行 206-212）

この箇所は簡易的なコピー & ペースト防止のためのプロシージャである。Word アプリケーションのウィンドウがアクティブになったイベントが捕捉された際に Class1 モジュールから呼び出され、本文中の先頭 1 単語をコピーしてクリップボードを上書きしている。

6.　カスタム XML 関連のプロシージャ（Module1：行 215-312）

カスタム XML を作成する、カスタム XML に追記する等の処理が記述されたプロシージャである。この部分については参考文献 3) の Web サイトに記載されたソースコードをそのまま流用している。

7.　ウィンドウアクティブイベント捕捉（Class1：行 2-12）

Word アプリケーションがアクティブになった際に自動的に呼び出されるプロシージャである。コピー & ペースト防止処理および他の Word ウィンドウを全て閉じる処理をしている。

8.　保存イベント捕捉（Class1：行 14-25）

Word VBA は保存イベント直前は BefreSave で捕捉可能だが、保存イベント後の AfterSave が定義されておらず、保存後の自動処理ができない仕様となっている。そのため、保存イベントを捕捉した際にカスタム XML 追記の AtSave プロシージャを呼び出し、おそらく処理が完了しているであろう 1 秒後に AfterSave プロシージャを呼び出す手順として実装している。

V.　既知の問題点

Word VBA の仕様による制限、および VBA のみで実装したことによる限界によって、次の問題点がある。

1.　Mac では入力できない 2 バイト文字が多い

Mac 版 Word において、入力時に特定の漢字が消えることがある。このバグは回避できないため、Mac ユーザーの学生に対してはひらがな入力するように指示しておくことが望ましい。

2.　はしご高などの異体字の入力で問題が発生する

VBA のソースコードが Shift-JIS であることによる仕様上の制限である。そのような漢字を氏名に含む学生には標準字体を用いるよう指導する必要がある。

3.　カスタム XML に保存時刻が記録されない場合が稀にある

稀にファイルを開いた時刻のみが記録され、保存時刻が記録されていない場合がある。おそらく処理速度が遅いパソコンにおいて発生すると推察するが、同じスペックの大学 PC で比較しても発生しないことがほとんどであり、再現性が伴わないため原因の追及が困難である。もっとも、約 300 名の受講生で 1% も発生しない現象であったこと、常に同じ学生の提出ファイルで発生していたことなどから、実用上は特に問題はなかった。

4.　上書き保存時に編集制限が掛からないことが非常に稀にある

提出ファイルを確認するために開いた際、編集制限が掛かっていなかった事例も確認されている。300 名 ×12 回の Word ファイル提出課題において 2 例が確認された。理由は不明である。

5.　協力者がいた場合の不正行為を排除できない

Word 不正行為防止マクロを使用したところで、「学籍番号及び氏名を協力者に伝えて自分の

代わりに課題をやってもらう」タイプの不正行為を防ぐことは困難であり、これが最大の問題点であるとも言える。防止策としては、「自分のパソコンまたは大学のパソコン以外で作成したファイルは受け付けない」「他人のパソコンで作成したファイルは受け付けない」と明示することで一定の効果はあるが、例えばパソコンごと協力者に貸し出されると不正行為を発見することは困難であり、本気の不正行為には無力であるとも言える。

一方で、安易に不正行為に手を染める学生の大半はそこまで面倒なことをせず、ただ単に誰かのファイルをもらってそのまま提出する傾向が非常に強い。実際にこの Word 不正行為防止マクロを半期の授業で使っていた中で、協力者に代わりにやってもらったことが疑われる事例は、筆者の観測する限りにおいては 1 例もなかった。協力者の可能性が排除できないにせよ、無料で個人が出来る不正行為対策としては運用上十分な効果が見込めるというのが、この Word 不正行為防止マクロのメリットである。

VI. まとめ

Word 不正行為防止マクロの導入によって、不正行為検出のための提出ファイルチェックの負担が大幅に軽減された。以前はファイルのプロパティで作成者をチェックするという、手間が掛かる割に効果の薄い方法でしか不正行為を見つける手段がなかったが、カスタム XML に学籍番号と氏名、さらには作業開始日時および保存日時を記録することによって、明白な不正行為は確実な証拠をもって発見できるようになった。ほとんどの学生は真面目に課題に取り組んでいるにも関わらず疑いの目をもって採点せねばならない心理的負荷から解放されたことは、筆者にとって非常に有用であった。

それに加えて、非常に簡易ではあるがコピー & ペースト防止機能をつけたことによって「ちょっとした出来心」のような段階で不正行為を防止できたことも推察される。学生の不正行為は基本的に出来心レベルのものであり、それさえ防止してしまえばほとんどの不正行為は撲滅できると考えられる。

また、不正行為防止以外の二次的な効用として、課題に取り組んだ実際の学習時間を自己申告よりも正確に把握できるメリットがあったことを明記しておきたい。課題ごとの学習時間を把握することで難易度調整に活用する、学習時間が非常に長い学生は習熟度に問題を抱えているからサポートするなど、授業改善につなげられる可能性があると言える。

以上のことから、大掛かりで費用の掛かるシステムを導入することなく、VBA マクロという簡便な手法によって、情報基礎科目における不正行為防止に効果が得られたこと、さらには正確な学習時間の把握によって効果的な授業改善の可能性が示唆されたことを結論としたい。

謝辞

Word 不正行為防止マクロの作成にあたって、駿河台大学情報処理教育センター内田講師には動作確認や不具合の報告等、多大な援助をいただきました。厚く御礼申し上げます。

別添資料

別添資料 1：Module1 ソースコード
別添資料 2：Class1 ソースコード
別添資料 3：作業ログ確認用マクロのソースコード（Normal.dotm 追記用）

別添したソースコードは OS に依存する API を一切使っていないため、Windows、Mac のどちらでも動作する。また、Word2013 以降のバージョンであれば動作確認は取れている。

参考文献

1) 巳波弘佳．オンライン授業における不正防止対策の取組みと展望．大学教育と情報．2022 年度 No.1，p.14-17
2) 今福啓．Excel のレポート課題における学業不正防止システムの提案．情報学研究．2018,

no.7，p.24-34
3) きぬあさ．“[Office VBA] カスタム XML を使って設定情報などをドキュメントに保存する方法”．初心者備忘録．2019．https://www.ka-net.org/blog/?p=11930，（参照 2022-09-30）．

別添資料 1：Modle1 ソースコード

```
'Word のコピペ防止マクロ Ver.3.06
' カスタム XML 関連のソースコードは、https://www.ka-net.org/blog/?p=11930（2021 年 4 月 1 日閲覧）を流用した
'Word ファイルの保護パスワードはハードコードしてある。YOURPASSWORD をすべてオリジナルのパスワードに変更すること。

Option Explicit

Private Const ns As String = "CL1"  'namespace
Private StartTime As String         ' カスタム XML の ID 用に作業開始時刻を保存するモジュール変数

Dim X As New Class1
Public NoSaveFlag As Boolean        ' カスタム XML を書き込まない上書き保存のフラグ変数。True で書き込まない。
Public CheckModeFlag As Boolean     ' 中身を確認するモードのフラグ変数。True で常に編集制限をかける
Public NoCopyFlag As Boolean     ' ヘッダー編集中のコピー回避フラグ。Inputbox を閉じたときに Window アクティブイベントを捕捉してしまうので。


Sub AutoOpen()
    Dim stime As String
    Dim userid As String
    Dim username As String
    Dim hdr As String
    Dim flag As Boolean

    'Word のイベント捕捉用クラスをセットする
    Set X.myapp = Word.Application

    ' 編集制限されているときにさらに編集制限するとエラーが出るので、そのエラーを無視する
    On Error Resume Next

    ' 明示的にアクティブにする
    ThisDocument.Activate

    ' 自動保存をオフにする
    'ThisDocument.AutoSaveOn = False
    'Word2013 では ActiveDocument じゃないとエラーになる
    ActiveDocument.AutoSaveOn = False

    ' 印刷レイアウトにする
    ActiveWindow.View.Type = wdPrintView

    ' フラグ変数の初期化
    NoSaveFlag = False
    CheckModeFlag = False
    NoCopyFlag = False

    ' 起動時に編集を制限して上書き保存する（フォーム以外入力できない状態にする）
    ThisDocument.Protect Type:=wdAllowOnlyFormFields, Password:="YOURPASSWORD"
```

```
47:
48:   'Save イベント捕捉プロシージャ回避フラグを立てる
49:   NoSaveFlag = True
50:   CheckModeFlag = True
51:
52:
53:   ' 編集制限状態で上書き保存する
54:   ThisDocument.Save
55:
56:   ' フラグを消す
57:   NoSaveFlag = False
58:   CheckModeFlag = False
59:
60:
61:   'InputBox で学籍番号を取得
62:   flag = False
63:   Do
64:       userid = InputBox(" 学籍番号（数字 7 桁）を入力してください ")
65:       userid = Trim(userid)
66:       If Len(userid) = 0 Then     'userid が長さが 0 なら確認モードになる
67:           CheckModeFlag = True
68:           Exit Sub
69:       ElseIf IsNumeric(userid) = True And Len(userid) = 7 Then    'userid が数字でかつ 7 桁であれば、ループ
抜けフラグを立てる
70:           flag = True
71:       End If
72:   Loop Until flag = True
73:
74:   'InputBox で氏名を取得
75:   username = InputBox(" 氏名を入力してください ")
76:   username = Trim(username)
77:   If Len(username) = 0 Then   'username が長さ 0 なら確認モードになる
78:       CheckModeFlag = True
79:       Exit Sub
80:   End If
81:
82:
83:   ThisDocument.Activate
84:   NoCopyFlag = True
85:
86:   ' 以下、ヘッダーに学籍番号と氏名を出力する
87:   On Error Resume Next    ' 編集制限されていないときに Unprotect を試みるとエラーになるので、それを
無視する
88:   ThisDocument.Unprotect ("YOURPASSWORD")  ' 編集制限の解除
89:
90:   ' ヘッダーに出力する文字列を結合する
91:   hdr = " 学籍番号：" & userid & "    氏名：" & username
92:
93:   ' ヘッダーを編集モードにする
94:   With ActiveWindow.View
95:       .Type = wdPrintView
96:       .SeekView = wdSeekCurrentPageHeader
97:   End With
```

```
 98:
 99:    ' ヘッダーをクリア
100:    Selection.WholeStory
101:    Selection.Delete
102:
103:    ' ヘッダーに学籍番号と氏名を入力
104:    Selection.TypeText Text:=hdr
105:    Selection.ParagraphFormat.Alignment = wdAlignParagraphRight
106:
107:    ' ヘッダーフッターツールを閉じる
108:    ActiveWindow.View.SeekView = wdSeekMainDocument
109:
110:    NoCopyFlag = False
111:
112:    ' 作業開始時刻を取得する
113:    stime = Now()
114:
115:    ' 学籍番号と ID と開始時刻を Custom XML に出力する
116:    Call AtOpen(stime, userid, username)
117:
118:End Sub
119:
120:' 閉じるときのイベントはコメントアウト
121:'Sub AutoClose()
122:'    ' 編集制限されているときにさらに編集制限するとエラーが出るので、そのエラーを無視する
123:'    On Error Resume Next
124:'
125:'    ' 終了時に編集を制限して上書き保存する（フォーム以外入力できない状態にする）
126:'    ThisDocument.Protect Type:=wdAllowOnlyFormFields, Password:="YOURPASSWORD"
127:'
128:'    'Save イベント捕捉プロシージャ回避フラグを立てる
129:'    NoSaveFlag = True
130:'
131:'    ' 編集保護状態で上書き保存する
132:'    ThisDocument.Save
133:'
134:'    ' フラグを消す
135:'    NoSaveFlag = False
136:'End Sub
137:
138:
139:Sub AtOpen(stime As String, userid As String, username As String)
140:    ' 作業開始時のカスタム XML 追加ルーチン
141:    ThisDocument.Activate
142:
143:    ' グローバル変数に今回の作業開始時刻を代入する（カスタム XML の ID 用）
144:    StartTime = stime
145:
146:    ' カスタム XML 追加
147:    SaveCustomProperty ThisDocument, ns, "ID " & StartTime, userid
148:    SaveCustomProperty ThisDocument, ns, "Name " & StartTime, username
149:    SaveCustomProperty ThisDocument, ns, "OfficeUser " & StartTime, Application.username
150:    SaveCustomProperty ThisDocument, ns, "starttime " & StartTime, StartTime
```

```
151:
152:    ' 保存
153:    NoSaveFlag = True
154:    ThisDocument.Save
155:
156:    NoSaveFlag = False
157:
158:End Sub
159:
160:
161:Sub AtSave()
162:    On Error Resume Next
163:    Dim endtime As String
164:    ThisDocument.Activate
165:
166:    'Save イベント捕捉プロシージャ回避フラグのチェック
167:    If NoSaveFlag = False Then
168:        'NoSaveFlag が立っていない場合は、保存直前の時刻を取得してカスタム XML 出力しておく
169:        endtime = Now()
170:
171:        ' カスタム XML に保存時刻を追加する
172:        SaveCustomProperty ThisDocument, ns, "endtime " & StartTime, endtime
173:    End If
174:
175:    'Save 時は必ず編集制限をする
176:    ThisDocument.Protect Type:=wdAllowOnlyFormFields, Password:="YOURPASSWORD"
177:
178:    ' 何もなければこの後で Save が行われるはず
179:End Sub
180:
181:
182:Sub AfterSave()
183:    ' セーブ 1 秒後に呼び出されるルーチン
184:    On Error Resume Next
185:    ThisDocument.Activate
186:
187:    While Word.ActiveDocument.Application.BackgroundSavingStatus <> 0
188:        ' セーブ中のキューが残っている場合は OS に処理を戻す。
189:        DoEvents
190:    Wend
191:
192:    ' 動作確認用 msgbox の残骸
193:    'MsgBox "AfterSave が呼び出されました "
194:
195:    'CheckModeFlag が True なら編集制限、False なら編集制限解除
196:    If CheckModeFlag Then
197:        ' 編集制限をかける
198:        ThisDocument.Protect Type:=wdAllowOnlyFormFields, Password:="YOURPASSWORD"
199:    Else
200:        ' 編集制限を解除する
201:        ThisDocument.Unprotect ("YOURPASSWORD")
202:    End If
203:
```

```
204:End Sub
205:
206:Sub CopyWords()
207:   ' ヘッダー選択時のエラー回避
208:   On Error Resume Next
209:
210:   ' 本文先頭の 1 単語をコピーする
211:   ThisDocument.Words(1).Copy
212:End Sub
213:
214:
215:Private Sub SaveCustomProperty(ByVal Doc As Object, _
216:                             ByVal ns As String, _
217:                             ByVal property_name As String, _
218:                             ByVal property_value As String)
219:' カスタムプロパティ設定
220:  Dim parts As Office.CustomXMLParts
221:  Dim part As Office.CustomXMLPart
222:  Dim root As Office.CustomXMLNode
223:  Dim target As Office.CustomXMLNode
224:  Dim target_attr As Office.CustomXMLNode
225:  Dim child As Object 'IXMLDOMElement
226:  Dim attr_id As Object 'IXMLDOMAttribute
227:  Dim attr_val As Object 'IXMLDOMAttribute
228:  Dim d As Object
229:
230:  Set parts = Doc.CustomXMLParts.SelectByNamespace(ns)
231:  If parts.Count < 1 Then
232:    Set part = InitCustomXMLParts(Doc, ns)
233:  Else
234:    Set part = parts.Item(1)
235:  End If
236:
237:  Set root = part.DocumentElement
238:  Set target = root.SelectSingleNode("//CustomProperty[@id='" & property_name & "']")
239:  If target Is Nothing Then
240:    'CustomProperty 要素
241:    'id 属性：property_name
242:    'value 属性：property_value
243:    Set d = CreateObject("MSXML2.DOMDocument.6.0")
244:    Set child = d.createElement("CustomProperty")
245:    Set attr_id = d.createAttribute("id")
246:    attr_id.NodeValue = property_name
247:    child.Attributes.setNamedItem attr_id
248:    Set attr_val = d.createAttribute("value")
249:    attr_val.NodeValue = property_value
250:    child.Attributes.setNamedItem attr_val
251:    d.appendChild child
252:    root.AppendChildSubtree d.XML
253:  Else
254:    For Each target_attr In target.Attributes
255:      If target_attr.BaseName = "value" Then
256:        target_attr.NodeValue = property_value
```

```
257:       Exit For
258:     End If
259:   Next
260: End If
261: 'If Not root Is Nothing Then Debug.Print root.XML ' 確認用
262:End Sub
263:
264:Private Function LoadCustomProperty(ByVal Doc As Object, _
265:                                   ByVal ns As String, _
266:                                   ByVal property_name As String) As String
267:' カスタムプロパティ読込
268: Dim parts As Office.CustomXMLParts
269: Dim root As Office.CustomXMLNode
270: Dim target As Office.CustomXMLNode
271: Dim target_attr As Office.CustomXMLNode
272:
273: Set parts = Doc.CustomXMLParts.SelectByNamespace(ns)
274: If parts.Count > 0 Then
275:   Set root = parts.Item(1).DocumentElement
276:   Set target = root.SelectSingleNode("//CustomProperty[@id='" & property_name & "']")
277:   If Not target Is Nothing Then
278:     For Each target_attr In target.Attributes
279:       If target_attr.BaseName = "value" Then LoadCustomProperty = target_attr.NodeValue
280:     Next
281:   End If
282: End If
283:End Function
284:
285:Private Sub DeleteCustomProperty(ByVal Doc As Object, _
286:                                 ByVal ns As String, _
287:                                 ByVal property_name As String)
288:' カスタムプロパティ削除
289: Dim parts As Office.CustomXMLParts
290: Dim root As Office.CustomXMLNode
291: Dim target As Office.CustomXMLNode
292:
293: Set parts = Doc.CustomXMLParts.SelectByNamespace(ns)
294: If parts.Count > 0 Then
295:   Set root = parts.Item(1).DocumentElement
296:   Set target = root.SelectSingleNode("//CustomProperty[@id='" & property_name & "']")
297:   If Not target Is Nothing Then target.Delete
298: End If
299: 'If Not root Is Nothing Then Debug.Print root.XML ' 確認用
300:End Sub
301:
302:Private Function InitCustomXMLParts(ByVal Doc As Object, _
303:                                   ByVal ns As String) As Office.CustomXMLPart
304:' カスタム XML 初期化
305: Dim d As Object, root As Object
306:
307: Set d = CreateObject("MSXML2.DOMDocument.6.0")
308: Set root = d.createElement("CustomProperties")
309: root.setAttribute "xmlns", ns
```

```
310:  d.appendChild root
311:  Set InitCustomXMLParts = Doc.CustomXMLParts.Add(d.XML)
312:End Function
313:
314:Private Sub DeleteCustomXMLPart(ByVal Doc As Object, _
315:                               ByVal ns As String)
316:' カスタム XML 削除
317:  Dim parts As Office.CustomXMLParts
318:
319:  Set parts = Doc.CustomXMLParts.SelectByNamespace(ns)
320:  If parts.Count > 0 Then parts.Item(1).Delete
321:End Sub
```

別添資料２：Class1 ソースコード

```
1:Public WithEvents myapp As Word.Application
 2:
 3:Private Sub myapp_WindowActivate(ByVal Doc As Document, ByVal Wn As Window)
 4:    'Word のウィンドウがアクティブになったイベントを捕捉して、本文の先頭 1 文字をコピーする
 5:    'Ctrl + V には有効だが、Windows＋V やクリップボード履歴には無力
 6:    'Word 内でのコピペは普通に使えるようにするための苦肉の策
 7:    If NoCopyFlag = True Then
 8:        Exit Sub
 9:    End If
10:
11:    Call CopyWords
12:End Sub
13:
14:Private Sub myapp_DocumentBeforeSave(ByVal Doc As Document, SaveAsUI As Boolean, Cancel As Boolean)
15:    ' 他の Word の保存イベントを捕捉していた場合は、何もしない
16:    If Doc.Name <> ThisDocument.Name Then
17:        Exit Sub
18:    End If
19:
20:    ' 保存イベントを捕捉したら、保存が実行される前にその時刻をカスタム XML に追加する
21:    Call AtSave
22:
23:    '1 秒後に編集制限解除のプロシージャを呼び出す
24:    Word.Application.OnTime Now() + TimeValue("00:00:01"), "AfterSave"
25:End Sub
```

別添資料 3：作業ログ確認用マクロのソースコード（Normal.dotm 追記用）

```
Sub カスタム XML チェック ()
'Word 不正行為防止マクロで記録された
'「学籍番号」「氏名」「Office User 名」「Open 日時」「Save 日時」
' などを表示するためのマクロです。

' 使い方
' このマクロを Word の normal.dotm の標準モジュールに貼り付けて利用してください。
' また、クイックアクセスツールバーの 1 番目に登録すると Alt+1 でチェックできるようになります。
'YOURPASSWORD を不正行為防止マクロ本体のパスワードと同一に変更してください。

    Dim ns As String
    ns = "CL1"

    Dim parts As Office.CustomXMLParts
    Dim part As Office.CustomXMLPart
    Dim root As Office.CustomXMLNode
    Dim target As Office.CustomXMLNode
    Dim target_attr As Office.CustomXMLNode
    Dim node_list As Object
    Dim msg As String

    ' 編集の制限を解除
    On Error Resume Next
    ActiveDocument.Unprotect ("YOURPASSWORD")

    ' カスタム XML チェック
    Set parts = ActiveDocument.CustomXMLParts.SelectByNamespace(ns)
    If parts.Count < 1 Then
        MsgBox (" 記録なし ")
    Else
        Set part = parts.Item(1)
        Set root = part.DocumentElement
        Set node_list = root.ChildNodes

        For Each target In node_list
            For Each target_attr In target.Attributes
                If target_attr.BaseName = "value" Then msg = msg & target_attr.NodeValue & vbCrLf
            Next
        Next
    MsgBox msg
    End If

End Sub
```