Correction automatisée de maquettes structures au format IFC via un script python

Culture Sciences de l'Ingénieur

Xavier JOURDAIN

Édité le 16/05/2019

école normale supérieure paris—saclay

Cette ressource est issue de la 5e édition EduBIM [1], *journées de l'enseignement et de la recherche sur la maquette numérique et le BIM en France qui se sont déroulées les 15 et 16 mai 2019 à l'ENS Paris-Saclay. EduBIM est le lieu de rencontres et d'échanges autour du BIM (Building Information Modeling) mettant en relation les enseignants, les chercheurs et les formateurs.*

Cette ressource détaille un atelier de 90 minutes réalisé le 16 mai 2019 auprès d'enseignants du secondaire et du supérieur amenés à former leurs étudiants à l'utilisation de maquettes numériques et afin de leur permettre de s'intégrer dans des démarches BIM.

Table des matières

RESSOURCE ..... ERREUR ! SIGNET NON DEFINI.

1. INTRODUCTION **1**

1.1. CONTEXTE 1
1.2. PRISE EN MAIN DE L'OUTIL 3
1.3. AU-DELA DE L'OUTIL 3
1.4. DEROULEMENT DE L'ATELIER 3

2. DESCRIPTION DU SCRIPT DE CORRECTION **3**

2.1. VERSION DE PYTHON ET D'IFC 3
2.2. OPTIONS D'EXPORT SOUS REVIT 3
2.3. LES VARIABLES D'ENTREE A MODIFIER PAR L'ENSEIGNANT·E 4
2.4. LA STRUCTURE DU SCRIPT 4
2.5. LES FICHIERS DE SORTIE 6

3. BIBLIOGRAPHIE **6**

1. Introduction

1.1. Contexte

Avec les nombreux projets de maquettes numériques à corriger, l'enseignant·e peut se retrouver à faire un travail de vérification fastidieux et peu enrichissant. Le temps étant précieux, nous proposons de faire une première correction via un script automatique écrit dans le langage python sur des maquettes au format IFC que les étudiants auraient à rendre. Cette démarche est

particulièrement adaptée à des modules d'auto-formation des étudiants leur permettant d'apprendre le fonctionnement d'un logiciel comme Revit ou Tekla par exemple (cf Figure 1).

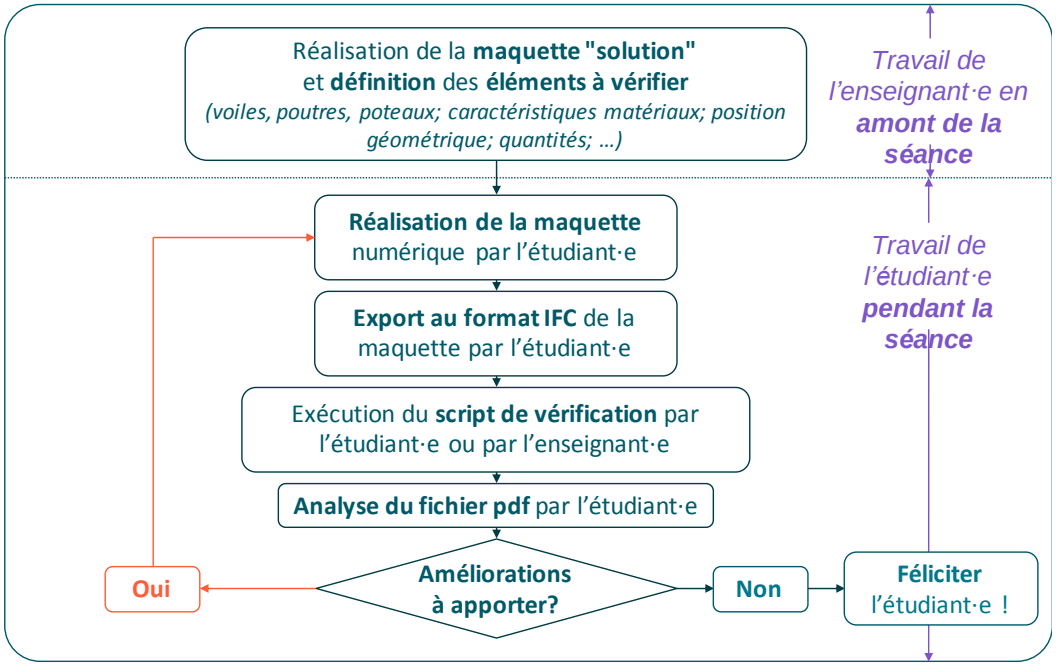


Figure 1: Organigramme de la correction automatisée en version Small Private Online Courses (SPOC) ou Massive Online Open Courses (MOOC)

L'autre possibilité est de l'utiliser plus classiquement pour évaluer un fichier rendu par un étudiant à la fin d'une séance (cf Figure 2).

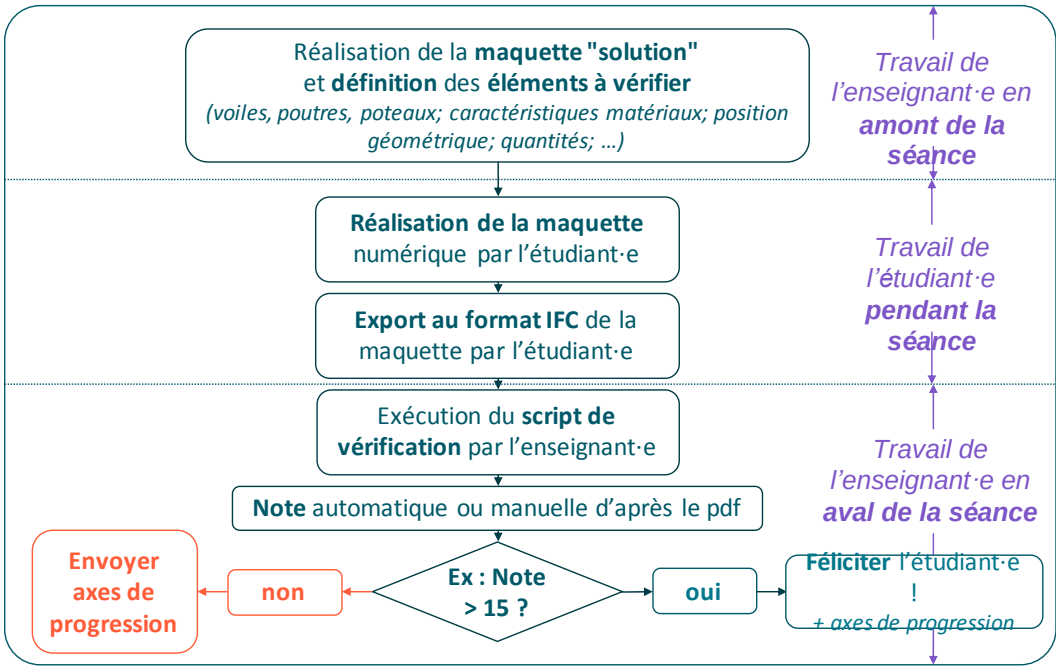


Figure 2: Organigramme de la correction automatisée en version maquette IFC rendu à l'enseignant

L'idée principale de cet outil est de passer moins de temps sur les corrections de formations de bases pour avoir plus de temps à consacrer à encadrer des projets originaux et réfléchir à la meilleure pédagogie possible !

1.2. Prise en main de l'outil

L'enseignant pourra adapter le script à ses objectifs pédagogiques, vérifier les quantités et caractéristiques qu'il juge importantes et sortir automatiquement un compte-rendu spécifiant où sont les différences observées. Avec un peu d'expérience, la plupart des erreurs classiques pourront être repérées et le script pourra être enrichi d'année en année sur un projet donné.

La version présentée ne fait que vérifier certaines informations liées aux murs mais la structure du script est prête à accueillir les éléments qu'un enseignant souhaiterait vérifier.

1.3. Au-delà de l'outil

Les étudiants plébiscitent les autoformations où la correction peut être instantanée. Ce genre de script peut répondre à cette problématique en permettant à l'étudiant de tout de suite tenter de se corriger en analysant le rapport qui lui est fourni.

1.4. Déroulement de l'atelier

Durée : 1h30

Logiciels utilisés :

- Anaconda [2]
  - Distribution libre et open source de langage de programmation dont python munit de l'environnement de développement spyder (= éditeur avec coloration syntaxique)
  - Version 1.8.7 avec python 3.6
  - Logiciel gratuit
  - page de téléchargement du logiciel : https://www.anaconda.com/distribution/
- Librairie python FPDF
  - Librairie python permettant de générer des document pdf
  - Version 3 (datant de 2007)
  - Page de téléchargement : https://github.com/reingart/pyfpdf
  - Vous pouvez aussi directement télécharger le fichier zip joint à cette ressource puis décompresser le dossier « fpdf » dans :
    - Sous windows : %ProgramData%\Anaconda3\Lib\fpdf
    - Sous mac OS X : /Users/VOTRE_IDENTIFIANT/anaconda3/lib/python3.6/fpdf

2. Description du script de correction

2.1. Version de python et d'IFC

Le script est écrit en python 3.6 [3] et est destiné à lire des maquettes numériques au format IFC2X3 [4]. Il utilise la librairie FPDF qui permet de générer des pdf proprement et facilement.

2.2. Options d'export sous Revit

Afin d'avoir un fichier IFC avec les informations souhaitées, bien préciser les options de sortie du fichier IFC :

- Pour REVIT => bien préciser « Exporter les quantités de base » dans l'onglet « Jeux de propriétés » (cf Figure 3)

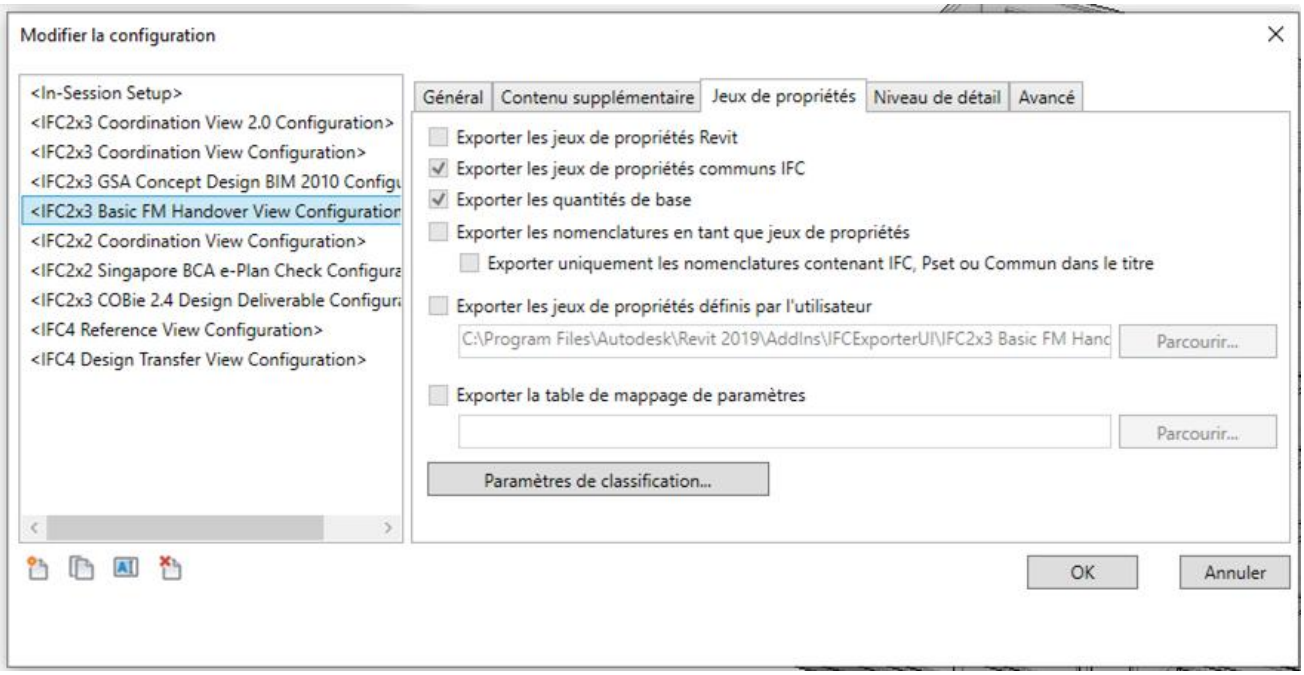


Figure 3: Type d'export à choisir sous Revit 2019 pour que le script fonctionne

2.3. Les variables d'entrée à modifier par l'enseignant·e

Quelques variables doivent être renseignées au départ du fichier afin de l'adapter à votre environnement notamment :

Nom du fichier IFC référence : **nom_du_fichier_IFC_Reference**
Dossier dans lequel se trouvent les fichiers étudiants : **dossier_fichiers_etudiants**
Sortie d'un bilan détaillé (**bilan_detaille**=True) ou synthétique (**bilan_detaille**=False)
Type d'export de Revit posant problème : **export_sans_quantites_IFC**

D'autres variables doivent être précisées si une note globale souhaite être mise à l'étudiant :

# poids sur chaque élément structurel
poids_des_elements={"Unités":0.5,"Niveaux":0.1,"Fondations":0,"Murs":5,"Dalles":0,"Poutres":0,"Poteaux":0}
# poids sur chaque quantité associée aux éléments à noter
poids_des_sous_elements_Fichier={"Unité de longueur":1,"Unité de surface":1,"Unité de volume":1}
poids_des_sous_elements_Niveau={"Nombre":1,"Altitudes":4}
poids_des_sous_elements_Mur={"Nombre":1,"Surf_banches":1,"Volume":1,"Niveau":0}
poids_des_sous_elements_Dalle={"Nombre":1,"Surface":1,"Volume":1,"Niveau":0.5}
poids_des_sous_elements_Poutre={"Nombre":1}
poids_des_sous_elements_Poteau={"Nombre":1}
poids_des_sous_elements_Fondations={"Nombre":1}

2.4. La structure du script

Description succincte du script

ToDoList des améliorations à mettre en place

Imports

Variables d'entrée du script (cf partie 2.3)

Constantes

Classes du programme

- Evaluation
- Structure
- File
- Storey
- Wall
- Element
- Fichier_FPDF

Fonctions

- accents_UTF8
- read_a_line
- **get_the_relationships_and_attributes**
- **read_a_file_and_get_all_the_elements**
- get_file_description
- get_specific_elements
- get_storeys
- get_walls_elements
- get_slabs_elements
- **get_structure_from_IFC**
- generate_pdf_from_one_structure
- relative_percentage
- print_comparison_bar
- print_comparison_result
- print_encouragement
- generate_pdf_for_comparison
- computing_final_note
- evaluate_from_a_comparison

Corps du programme

```
# Création d'un dictionnaire qui regroupera les données des fichiers IFC lus
dico_Structures = {}

# Données du fichier Référence
print("Analyse du fichier référence : {}".format(nom_du_fichier_IFC_Reference))
dico_Structures[nom_du_fichier_IFC_Reference] = get_structure_from_IFC(nom_du_fichier_IFC_Reference)

# Génération d'un fichier pdf à partir des éléments structurels d'un fichier IFC
generate_pdf_from_one_structure(dico_Structures[nom_du_fichier_IFC_Reference])

# Lecture et analyse des fichiers du dossier étudiants
dico_Evaluations = {}
for element in os.listdir(dossier_fichiers_etudiants):

    if not ((dossier_fichiers_etudiants[-1] == "/") or (dossier_fichiers_etudiants[-1] == "\\")):
        dossier_fichiers_etudiants += "/"

    if element.endswith('.ifc'):
        print("Analyse du fichier {}".format(element))

        dico_Structures[element] = get_structure_from_IFC(element,dossier_fichiers_etudiants)
```

```
        dico_Evaluations[element] = evaluate_from_a_comparison
                            (dico_Structures[nom_du_fichier_IFC_Reference],dico_Structures[element])
        dico_Structures[element] = generate_pdf_for_comparison
(dico_Structures[nom_du_fichier_IFC_Reference],dico_Structures[element],dico_Evaluations[element])
```

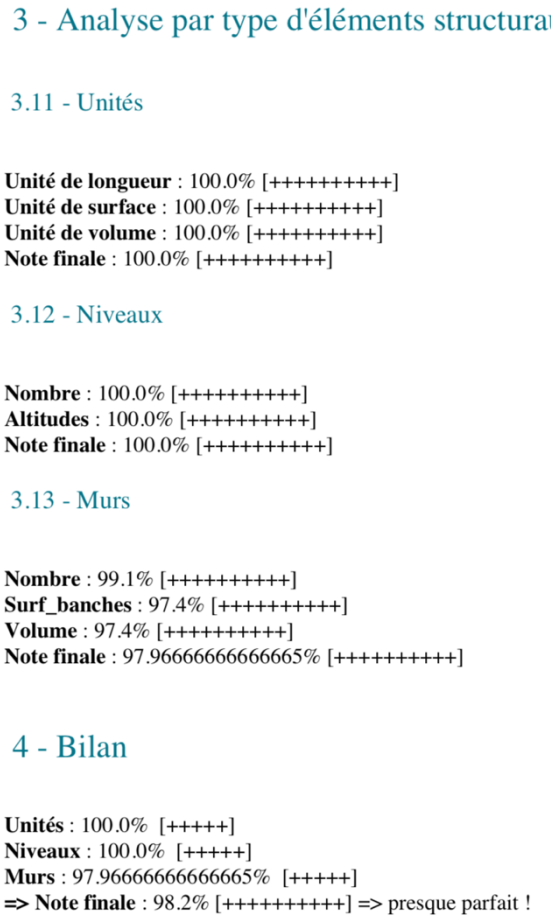
2.5. Les fichiers de sortie

Un fichier pdf faisant un bilan du fichier référence (= fichier correction de l'enseignant·e) :

1. Informations générales
2. Bilan par niveau

Un fichier pdf par document élève rendu (cf extrait sur la *Figure 4*) :

1. Informations générales (évaluation des unités choisies par exemple)
2. Analyse par niveau (évaluation du nombre de niveau, du nombre de murs, etc.)
3. Analyse par type d'éléments structuraux (unités, niveaux, murs, etc.)
4. Bilan
    - Évaluation par type d'éléments structuraux
    - Note finale avec la pondération choisie



3 - Analyse par type d'éléments structuraux

3.11 - Unités

Unité de longueur : 100.0% [++++++++++]
Unité de surface : 100.0% [++++++++++]
Unité de volume : 100.0% [++++++++++]
Note finale : 100.0% [++++++++++]

3.12 - Niveaux

Nombre : 100.0% [++++++++++]
Altitudes : 100.0% [++++++++++]
Note finale : 100.0% [++++++++++]

3.13 - Murs

Nombre : 99.1% [++++++++++]
Surf_banches : 97.4% [++++++++++]
Volume : 97.4% [++++++++++]
Note finale : 97.96666666666665% [++++++++++]

4 - Bilan

Unités : 100.0% [+++++]
Niveaux : 100.0% [+++++]
Murs : 97.96666666666665% [+++++]
=> Note finale : 98.2% [++++++++++] => presque parfait !

Figure 4: Exemple d'extrait de pdf de comparaison

3. Bibliographie

[1] EduBIM 2019, 15-16 05 2019. [En ligne]. Available: https://edubim2019.sciencesconf.org.

[2] Anaconda, «Page de téléchargement du logiciel Anaconda,» [En ligne]. Available: https://www.anaconda.com/distribution/. [Accès le 04 2019].

[3] Python, «page de téléchargement de python,» [En ligne]. Available: https://www.python.org/downloads/. [Accès le 04 2019].

[4] BuildingSMART IFC2x3, [En ligne]. Available: http://www.buildingsmart-tech.org/specifications/ifc-view-definition/coordination-view-v2.0/summary.

Les soutiens d'EduBIM 2019

Ressource publiée sur Culture Sciences de l'Ingénieur : http://eduscol.education.fr/sti/si-ens-paris-saclay