ÉCOLE CENTRALE LYON

Compte-rendu

PLM - maquettage numérique

Gestion de BOM sous CATIA

Groupe

LASSUS Thibaud

LEPAIS Corentin

Table des matières

1. Contexte

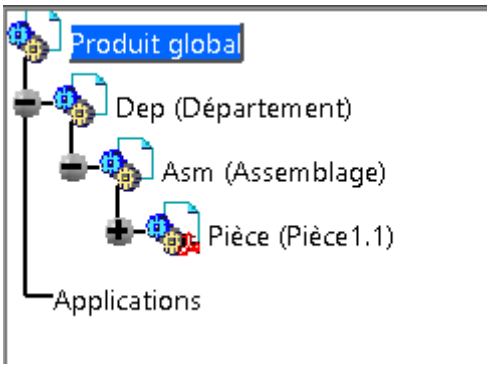
a. Expression du besoin

La génération de BOM sous Catia est possible, mais très peu ergonomique, surtout lorsque l'on travaille avec des gros assemblages (1500 pièces). L'idée ici, c'est d'utiliser les fonctionnalités plus avancées d'Excel (onglets, couleurs) pour rendre le BOM lisible. Il y a également certaines informations qu'il faut récupérer, comme le responsable de la pièce par exemple.

Un autre aspect sur lequel on veut travailler est la lisibilité des assemblages : quand on travaille sur une pièce, on a souvent envie de voir la référence et le nom (pour une pièce epsa, par exemple EN_03001 – Moteur). Lorsqu'on l'insère dans un assemblage, c'est cette phrase qui sert de base au nom de la pièce dans l'assemblage. Par défaut, ce sera : EN_03001 – Moteur (EN_03001 – Moteur.1). On arrive très vite à des assemblages qui sont illisibles, si chacun ne prend pas la responsabilité de renommer ses pièces. L'idée ici, c'est d'afficher plutôt EN_03001 (Moteur.1).

b. Structure de travail

La structure de l'assemblage est la suivante :



On a donc :

- Produit global (.CATProduct)
    - Départements (.CATProduct)
        - Assemblages (.CATProduct)
            - Pièces unitaires (.CATPart ou .CATProduct)

Concernant les dossiers, on va également être sur une structure fixe :

- Les pièces unitaires (qu'elles soient en CATPart ou CATProduct) sont placées dans le dossier de l'assemblage auquel elles appartiennent.
- Les départements et assemblages sont placés dans un sous-dossier de leur parent, et ce dossier porte leur nom.

2. Solutions mises en place

a. Création des éléments Catia

Lorsque l'on demande à toute une équipe de saisir des informations complètes sur chacune des pièces, on se retrouve très souvent avec des gens qui oublient, ou qui ne remplissent pas les bons champs.

Afin de palier à ce problème, on met en place 2 formulaires, plus explicites, qui vont récupérer les informations nécessaires, et les compléter dans les bons champs. On a ainsi :

- Numéro de pièce : champ *Référence*
- Nom (en français) : *Définition*
- Responsable : champ personnalisé, *respo*
- Provenance (acheté/fabriqué), pour les pièces unitaires : *Source*

Le principe est très simple : 2 fonctions, *nouvellePart* et *nouveauProduct* qui ouvrent chacune leur formulaire, et créent le fichier demandé, avec tous les champs pré-saisis. Cela permet un gain de temps considérable, surtout qu'entre 2 itérations les champs sont pré-enregistrés.

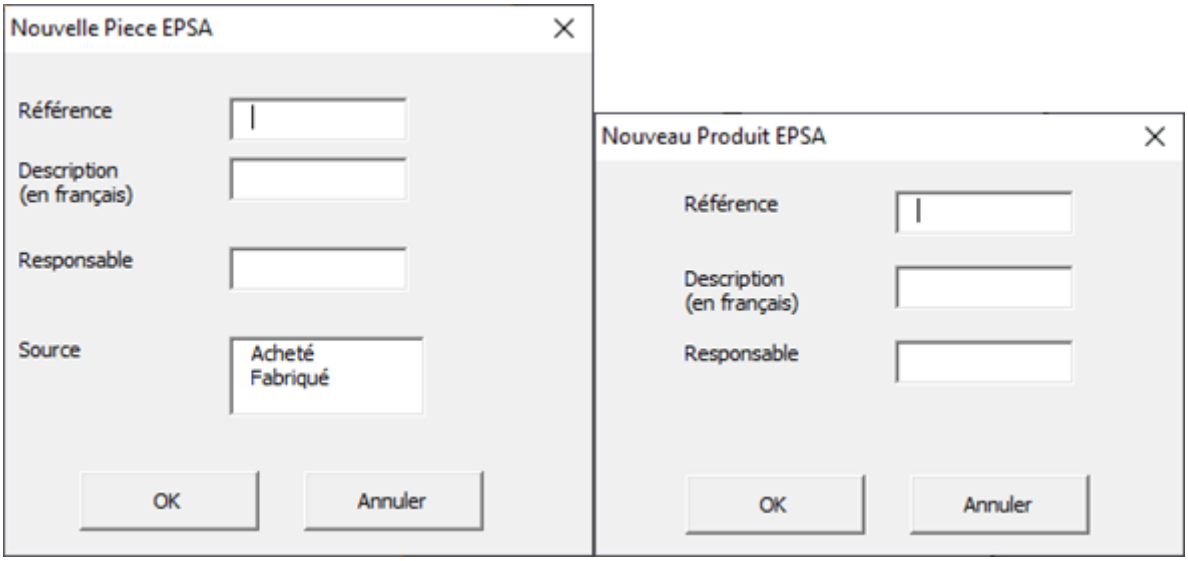


Figure 1 : Fenêtres de création de pièce ou de produit CATIA

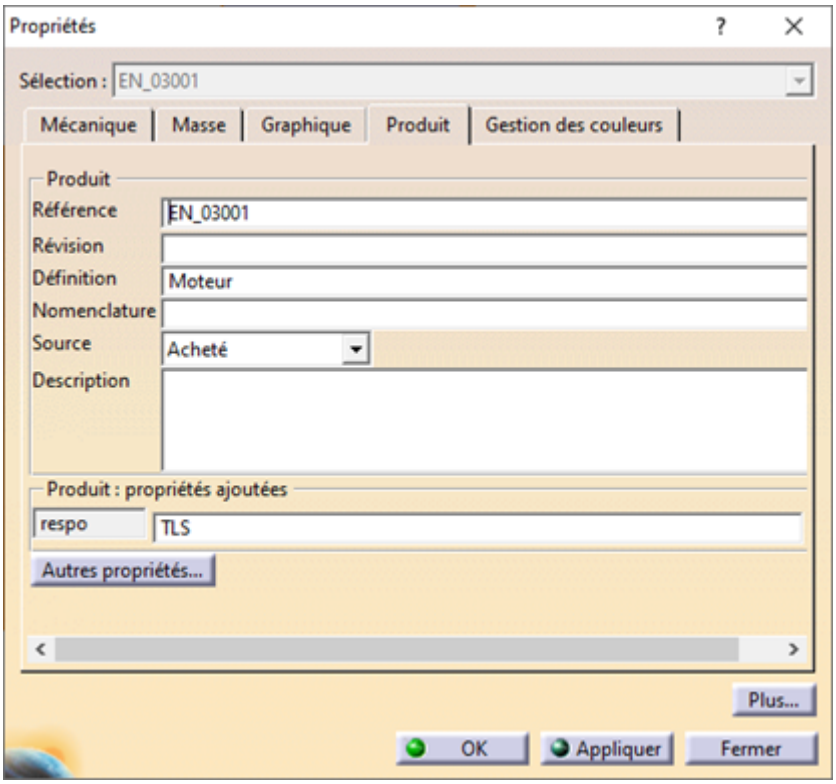


Figure 2 : Feuille de propriétés complétée

b. Insertion dans l'assemblage

Comme évoqué plus tôt, lors de l'insertion d'un élément dans un assemblage, le nom qui en résulte est peu lisible. On a donc, avec les informations décrites juste au-dessus, la possibilité d'obtenir la syntaxe : Référence (Definition.1).

La macro *insereComposant* va jouer ce rôle. De la même manière que dans l'insertion classique de Catia, on clique sur un produit, puis une fenêtre s'affiche pour choisir le composant à ajouter. La fonction *insere* est déportée dans le module *Fonctions* car elle est utilisée dans d'autres Macros.

Le cas où plusieurs exemplaires du même élément sont insérés est également prévu : comme le ferait Catia, on prend le premier numéro libre.

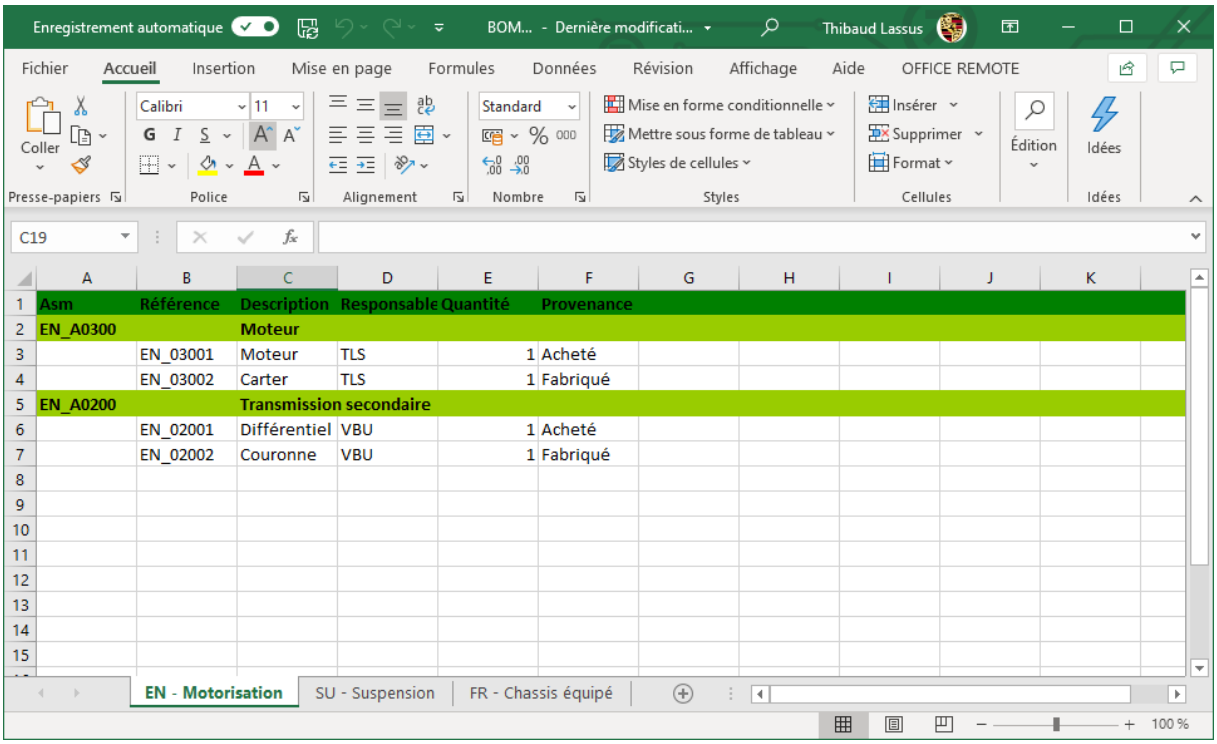
c. Génération du BOM

Pour le BOM, afin de maximiser la lisibilité, on va procéder comme suit :

- Chaque département correspond à un onglet Excel
- La première ligne d'un onglet correspond à l'en-tête, avec les intitulés des champs
- Les pièces sont listées sous l'assemblage auquel elles appartiennent
- Les assemblages sont listés les uns à la suite des autres
- La 1ère colonne permet l'identification de l'assemblage,
- La 2ème l'identification de la pièce.

Pour plus de lisibilité, on peut également utiliser des couleurs, du gras…. Dans le cas de l'EPSA, le nom des départements est fixé : quelque soit l'ordre dans la maquette, les couleurs sont attribués en fonction de la référence du département. Dans un cas générique, on pourrait affecter les couleurs en fonction de l'ordre dans lequel les départements sont placés.

Le fichier final donne :



Asm	Référence	Description	Responsable	Quantité	Provenance
EN_A0300		Moteur			
	EN_03001	Moteur	TLS	1	Acheté
	EN_03002	Carter	TLS	1	Fabriqué
EN_A0200		Transmission secondaire			
	EN_02001	Différentiel	VBU	1	Acheté
	EN_02002	Couronne	VBU	1	Fabriqué

Figure 3 : BOM généré par la macro

Principe de fonctionnement :

- On parcourt l'assemblage avec 3 boucles "For" imbriquées, pour passer en revue tous les départements, assemblages et pièces
- Pour l'attribution des couleurs, on utilise 2 dictionnaires (un pour chaque ton) : si le département est référencé, on lui donne sa couleur. Sinon, une couleur générique, intitulée "Default" (en gris).
- Dans le parcours de la feuille, on utilise un compteur *ligne*, qui est incrémenté à chaque fois que l'on passe au composant suivant.
- Pour chaque assemblage et pièce, on procède comme suit :
    - On vérifie s'il n'a pas déjà été passé en revue, à l'aide d'un dictionnaire. L'information stockée est la ligne à laquelle il a été placé.
    - Si on a déjà vu ce composant, on incrémente la colonne "*Quantité*" associée.
    - Sinon, on remplit les colonnes avec les informations disponibles, et on continue.
    - Si nécessaire, on fait la mise en forme de la ligne.
    - Et on ajoute l'élément dans le dictionnaire associé.
    - Lorsque l'on a fini de parcourir toutes les pièces d'un assemblage, on réinitialise le dictionnaire des pièces (de même pour le dictionnaire d'assemblages, lorsque l'on change de département).

d. Génération des CATProduct à partir du BOM

A partir de l'Excel généré précédemment, on recrée tous les product (global, département et systèmes)

- Une boucle for pour parcourir les départements <-> onglets de l'Excel
- En revanche, on ne sait pas combien d'assemblages il y a. On utilise donc une boucle while, avec comme limite la dernière ligne (remplie) de la feuille.

- Cette boucle est chargée de détecter les assemblages. Une 2ème à l'intérieur s'occupe des pièces. A chaque fois, elle teste l'existence de la pièce en question sous les formats CATPart et CATProduct, et insère celui qui existe.
- L'insertion est soumise à une boucle "for", qui répète l'opération autant de fois que nécessaire.
- L'enregistrement et la fermeture de chaque assemblage créé, sauf le global.

On obtient exactement le même arbre que lors de l'export (seul l'ordre peut être modifié, lorsque plusieurs instances d'un même composant sont insérées).

Cette fonction réinitialise la position des éléments. Une solution permettant de les remettre à la bonne place a été essayée, les problèmes rencontrés sont présentés plus bas.

e. Accès facilité

Pour une utilisation au quotidien, ces macros peuvent être présentées sous forme de barre d'outils sous Catia :

Cependant, nous n'avons pas trouvé comment exporter/importer les barres d'outils.

3. Problèmes rencontrés

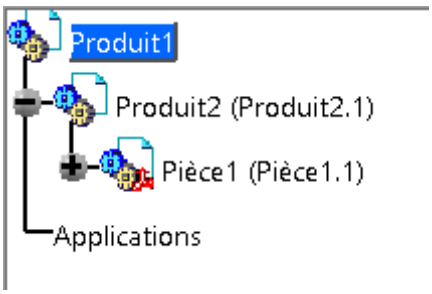
a. Navigation dans les assemblages

La structure est assez peu intuitive, entre Document, PartDocument, ProductDocument, Product et Part. Au début, beaucoup de recherches "à tâtons" pour arriver à récupérer les objets sur lesquels on veut travailler. La navigation dans le fichier d'aide de catia est peu intuitive également, le menu "ObjectBrowser" de VBA (touche F2) est assez pratique. On n'a pas les détails des aides sur les fonctions, mais une compréhension aisée du lien entre les types et les fonctions.

La sélection de l'élémént dans l'arbre (pour la fonction d'insertion) a également posé problème : l'aide sur les méthodes *SelectElement* est mal expliquée. Il ne faut pas se contenter de ce qu'on trouve dans l'aide Catia et aller voir des exemples d'utilisation, notamment pour la string renvoyée (qui permet d'éviter une erreur si la sélection est annulée par exemple.

b. Insertion dans les sous-assemblages

Lorsque l'on veut insérer un composant dans un sous-assemblage de l'objet de travail, la version initiale de la fonction n'arrivait pas à faire le bon rennomage. (ie. Sur *Pièce1* dans la capture ci-dessous).



J'ai d'abord pensé qu'il s'agissait d'un problème lié au mode de sélection du produit, d'où la fonction *InsereComposant2*, qui au lieu d'ajouter le composant puis de le selectionné, elle l'ouvre puis l'insère. Le problème était toujours le même.

Nous avons fini par trouver la solution, il fallait mettre *Set Assembly = Assembly.ReferenceProduct*, soit définir l'assemblage (*Produit2* ici) comme objet de travail.

c. Récupération des propriétés utilisateur

Lorsque l'on veut écrire les propriétés utilisateur dans le fichier Excel (ici, le responsable), le nom à placer dans la méthode "Item" n'est expliqué nulle part

```
Set piece = piece.ReferenceProduct
oExcel.Cells(ligne, 2) = piece.PartNumber
oExcel.Cells(ligne, 3) = piece.Definition
oExcel.Cells(ligne, 4) = piece.UserRefProperties.Item("Propriétés\respo").ValueAsString
oExcel.Cells(ligne, 5) = 1
oExcel.Cells(ligne, 6) = production(piece.source)
```

Le fichier d'aide n'indique que la possibilité d'utiliser l'indice de la propriété, ce qui peut être compliqué si un utilisateur rajoute des propriétés et modifie l'ordre.

o Func **Item**(CATVariant `iIndex`) As CATIAParameter

Retrieves a parameter using its index or its name from the from the Parameters collection.

Parameters:

`iIndex`

The index or the name of the parameter to retrieve from the collection of parameters. Count. As a string, it is the name you assigned to the parameter using the AnyObject.Name property or when creating the parameter.

Example:

This example retrieves the last parameter in the `parameters` collection:

```
Set lastParameter = parameters.Item(parameters.Count)
```

L'autre aspect qui n'était pas mentionné est le fait qu'au-delà de sélectionner *UserRefProperties* sur la pièce en cours, il faut placer un *Set piece = piece.ReferenceProduct*, sinon la commande ne marche pas.

d. Fichier excel déjà ouvert

Dans le module *lireBOM*, la première instruction est d'ouvrir le fichier du BOM. Sauf que le module CATIA bloque l'opération si le fichier est déjà ouvert.

Nous avons donc essayé d'utiliser la fonctionnalité équivalente chez excel (Application.Dialog), qui fonctionne sans erreur. Mais si le fichier était déjà ouvert, le programme ne le chargait pas correctement, et rencontrait assez vite une erreur.

Nous avons donc laissé le mode d'ouverture de Catia, plus restrictif mais qui fonctionne à 100%.

e. Manipulation du chemin de fichier

Ici on est plus proche de l'anecdote : dans le module *lireBOM*, lorsque l'on veut récupérer le chemin d'accès au dossier, il paraissait plus simple de demander à Excel le dossier dans lequel se trouve le BOM. Sauf que celui-ci était sur un dossier partagé OneDrive, et au lieu de renvoyer l'adresse en local sur le PC, on avait une URL correspondant au fichier en ligne, donc impossible de trouver le reste des fichiers.

Heureusement le mode Debug a permis de voir l'origine du problème assez vite (la valeur des variables s'affiche quand on passe la souris dessus).

On a donc récupéré le chemin à partir de l'adresse du fichier Excel, en éliminant le nom de fichier :

```
dossier = Left(File, InStrRev(File, "\") - 1)
```

(On conserve uniquement ce qui est à gauche du dernier "\")

f. Récupération des coordonnées

Sur ce point, nous n'avons pas réussi à trouver de solution. Il s'agit de faire un export en vue d'une reconstruction qui conserve la position. La fonction que nous voulons utiliser (et la seule trouvée sur internet) est la méthode *GetComponents*, qui s'applique ainsi :

o Sub **GetComponents**(CATSafeArrayVariant `oAxisComponentsArray`)

Returns the components of an object's position. This returns the 3D-axis system associated with the object.

Parameters:

`oAxisComponentsArray`

The array used to store the twelve components retrieved from the objet's position. The first nine represent succcessively the components of the x-axis, y-axis, and z-axis. The last three represent the coordinates of the origin point.

Example:

This example retrieves in `oAxisComponentsArray` the 3D-axis system components from the `Position` object associated with `MyObject`:

```
Dim oAxisComponentsArray ( 11 )
MyObject.Position.GetComponents oAxisComponentsArray
```

On récupère les 12 composantes de la position de l'objet (les coordonnées des vecteurs du repère ainsi que l'origine). L'idée est de les stocker dans le fichier Excel (dans lequel on ne combinerait pas les doublons, mais on écrirai chacun avec sa position) pour pouvoir les replacer correctement, avec *SetComponents*

o Sub **SetComponents**(CATSafeArrayVariant `iAxisComponentsArray`)

Sets the components of an object's position. This sets the 3D-axis system associated with the object.

Parameters:

`iAxisComponentsArray`

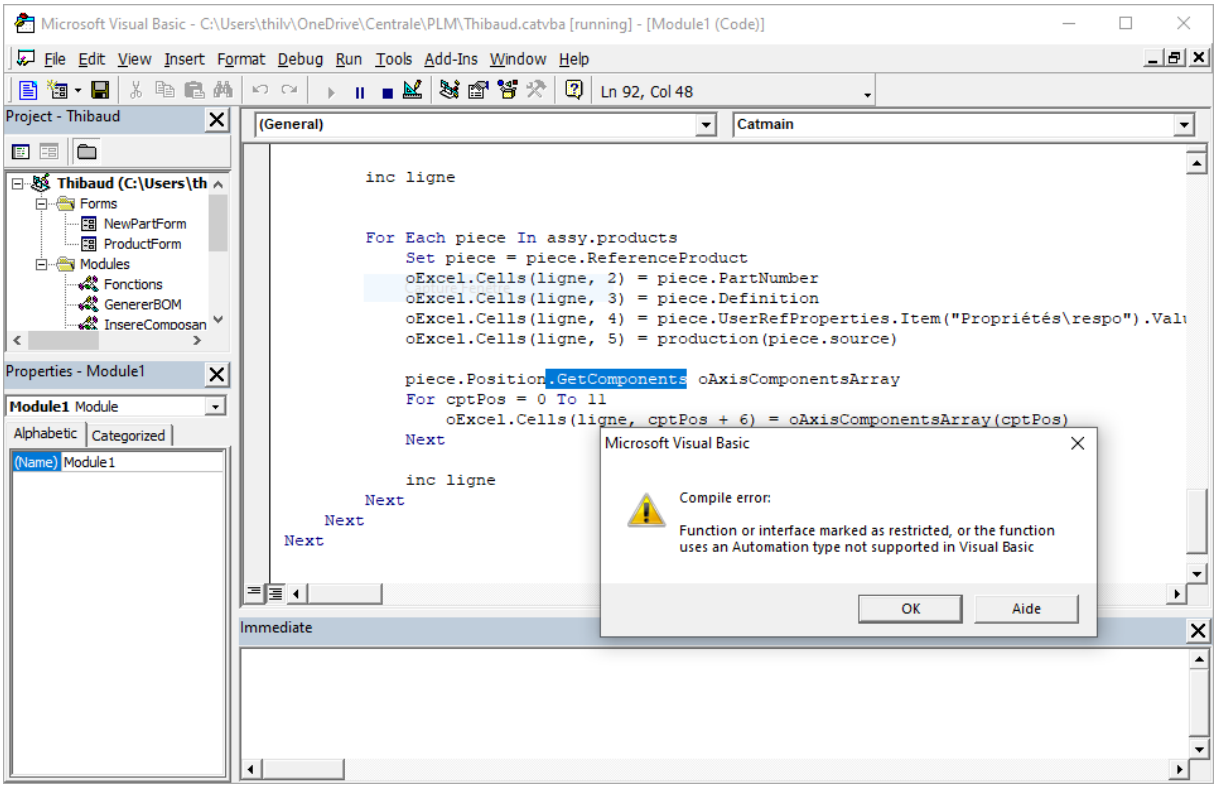
The array initialized with the components to set to the object's position. The first nine represent succcessively the components of the x-axis, y-axis, and z-axis. The last three represent the coordinates of the origin point.

Example:

This example sets the 3D-axis system components stored in `iAxisComponentsArray` to the `Position` object associated with `MyObject`:

```
Dim iAxisComponentsArray( 11 )
' x axis components
iAxisComponentsArray( 0 )  = 1.000
iAxisComponentsArray( 1 )  = 0
iAxisComponentsArray( 2 )  = 0.707
' y axis components
iAxisComponentsArray( 3 )  = 0
iAxisComponentsArray( 4 )  = 0
iAxisComponentsArray( 5 )  = 0.707
' z axis components
iAxisComponentsArray( 6 )  = 0
iAxisComponentsArray( 7 )  = -0.707
iAxisComponentsArray( 8 )  = 0.707
' origin point coordinates
iAxisComponentsArray( 9 )  = 1.000
iAxisComponentsArray( 10 ) = 2.000
iAxisComponentsArray( 11 ) = 3.000
MyObject.Position.SetComponents iAxisComponentsArray
```

Ces deux fonctions renvoient systématiquement une erreur au moment de la compilation :



Même en essayant sur un programme et assemblage basique pour du debuggage, l'erreur persiste, et aucune mention de ce problème (pour cette fonction) sur internet.

Il s'agit peut-être d'un bug sur mon installation Catia, j'ai laissé la fonction dans Module1, si vous voulez la tester.

Bilan

Hormis le dernier aspect, tous les modules fonctionnent correctement. Le BOM est très agréable à utiliser. Sur le modèle de ceux présentés, on peut imaginer bien d'autres paramètres personnalisés (fournisseur, état d'avancement de la conception…). Il s'agit ici d'une version plus de "démo", et totalement personnalisable.

Il est dommage que la partie Coordonnées ne fonctionne pas, si vous y arrivez ou avez une autre méthode, nous sommes preneurs.

On peut également imaginer des variantes à ce programme : par exemple, à partir du BOM, mettre à jour les informations des pièces et assemblages (si on veut changer les responsables de 15 pièces, cela va plus vite sur Excel que sur Catia)