**上位机配置总结**

前言

本文适合零基础配置artop和上位机，有xsd，uml基础的，准备知识部分可以直接略过。

# 目标

上位机的配置过程总体目标是**将TPS中的UML图和模型定义的内容，在artop上进行配置，转换成arxml文件，再根据生成的arxml文件和xsd文件，书写json，最终上位机加载arxml和json文件，完成建模方法在上位机上配置实现**。

# 准备知识

自学习一下XSD文件相关的内容，以下是参考链接，也可以自行上网查找

<https://www.w3school.com.cn/schema/schema_simple_attributes.asp>

如果对UML图不了解，了解一下UML图的六种关系，参考链接如下

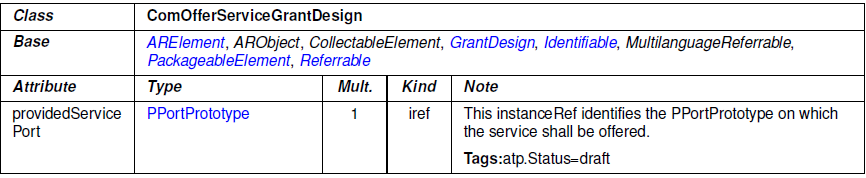
<https://www.cnblogs.com/shindo/p/5579191.html>

磨刀不误砍柴工，以上温习一下，对后面的理解有好处。

# 开发过程

## 1 TPS理解

**整个开发过程都是围绕着对TPS中需求的理解来进行的，TPS理解透彻了，工作完成了一半。**TPS中的UML图标识了各个模块的关系，其中建模表格说明了具体模块的关系。以下简单说明表格中各个项的含义。



**Class**：类名，当前类名称。

**Base**：当前类所引用的基类。

**Attribute**：属性，当前类的属性。

**Type**：类的类型，如定义int a，a的类型是int，以上图为例，providedServicePort的类型是PPortPrototype。

**Mult**：是指在上位机上元素可创建子节点的格个数。

0..1，可有可无，最多一个。

1，有且仅有一个。

0..\*，可有可无，可以有多个。

1..\*,必须存在，可以有多个。

**Kind**：iref ，表示引用关系。

attr，属性，可以是当前类的一个方法，属于包含关系。

**Note**：解释说明。

## 2 Artop配置

此处的配置是在将TPS文件中的UML图和类表的说明结合到一起，通过Artop的配置转换成arxml文件，arxml文件就是artop配置的最后成果。

Artop本身包含了TPS中各个元素，最好是将svn中已有的Neusar.arxml导入到Artop，然后再在原有的基础上进行配置，因为有很多基础原始已经有人配置好了，就不用重复配置了。

配置请直接在NeuSAR节点开始配置。

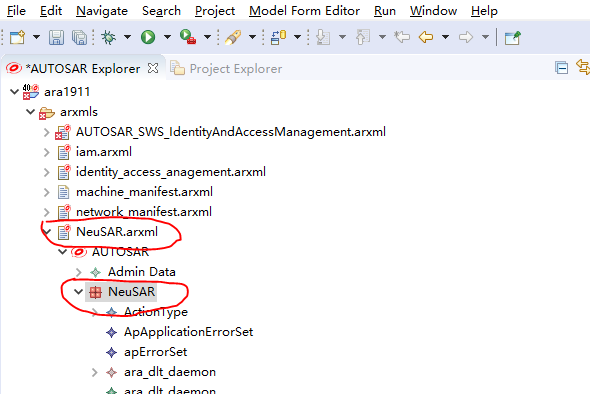
### 2.1 Artop创建工程和文件

先创建工程：file->new->AUTSAR Project。填上工程名称，finish。

在新建的工程上右键：new->AUTSAR file。Finsh。

以上创建一个默认的arxml，default.arxml。然后将Neusar.arxml放到default.arxml所在目录即可。刷新以下artop工程就可以看到Neusar.arxml。然后再在此基础上进行配置。

下图是导入Neusar.arxml文件后的结果。



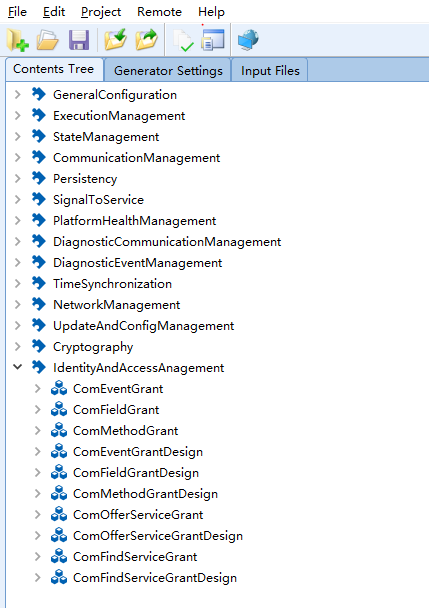
### 2.2 TPS中相关类的配置

右击NeuSAR->New Child->Elements，就可以看到TPS中有的元素名称。然后根据TPS中的UML图相关联关系，进行对应的配置。重点还是在于对TPS的理解。

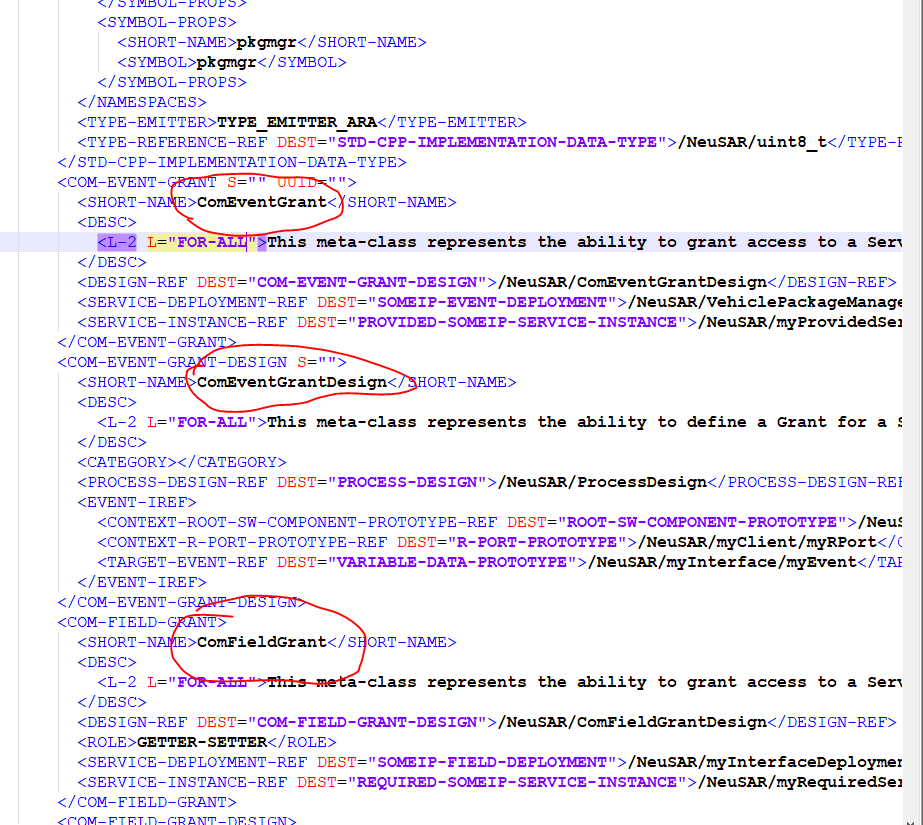
## 3 arxml到json文件的转换

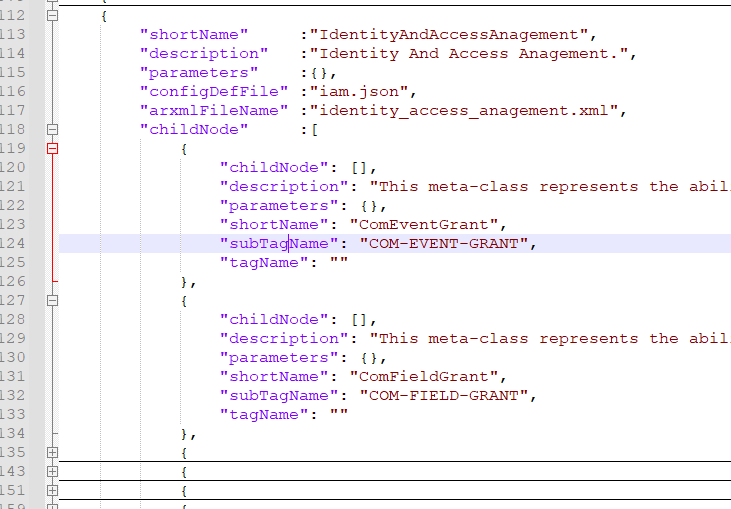
Artop配置完成后，将对应的arxml文件和xsd(ConfigurationToolEXE\generator\config\XMLSchema)文件导出来，打开可以通过短名查找自己配置的节点。接下来的工作便是将配置好的节点转换成json格式，进而能在上位机上进行加载，用于上位机的功能配置。在配置之前可以先打开上位机，对其后台的json文件和arxml文件初步查看了解以下。

### 3.1 主节点json配置



上位机上的这些节点，都是在json文件中配置出来并由上位机显示，如果是新增大一级节点或二级节点，就首先应当对treeConfig.json进行配置，在其中增加节点。一级节点的名称是模块的名称，比如IAM，一级节点的名称命名为IdentityAndAccessAnagement，二级节点名称，是在artop上配置的节点短名。二级节点的短名配置参考artop生成的Neusar.xml文件。





shortName：短名，一级二级节点的界面显示名称。

description：描述信息。

Parameters:参数，一二级节点一般没有参数，可以不填。

ConfigDefFile:引用的json配置文件名称。在配置完树一级二级节点后，还要做模块的详细配置，此处填写模块的json文件名称。

arxmlFileName：在上位机上配置节点也会生成arxml文件，此处填写一个armxl文件名称，用于后续配置存储的文件。

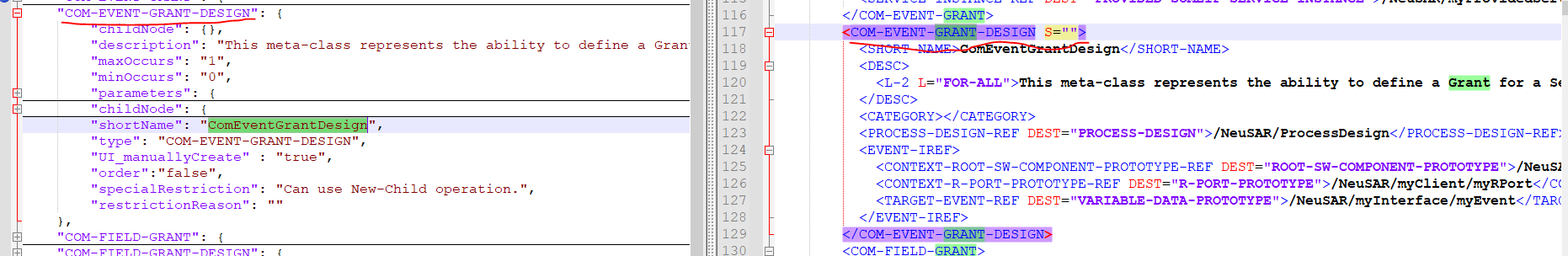
ChildNode：用于创建当前节点的子节点。

在配置完treeConfig.json后，重启上位机，json文件格式正确的情况下，新增的节点就会显示出来。

### 3.2 模块json配置

当模块节点生成后，就要配置子节点的具体属性和参数，在ConfigurationToolEXE\configFile中新创建一个json文件，参考Neusar.arxml，并按照treeConfig.json的二级节点的短名命名各个节点来进行具体的配置。

此处以IAM的ComEventGrantDesign为例来说明模块json的配置。

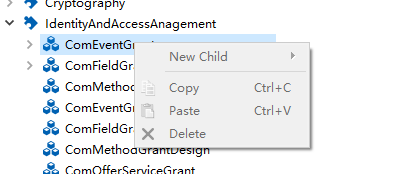


根据arxml文件的内容，在COM-EVENT-GRANT-DESIGN{ }中定义每一个节点的具体的属性。

descrption:描述

UI\_manuallyCreate：表示是否在界面上可以创建子节点，

false：右击节点不能创建子节点，“New Child”为灰色。



true：当前节点可以创建子节点。

maxOccurs：当UI\_manuallyCreate为true时，此处规定可创建最大节点的个数。”-1”表示可以创建任意多个子节点。

minOccurs：当UI\_manuallyCreate为true时,此处规定可创建最小的节点个数。

shortname：短名，跟arxml中的SHORT\_NAME一致。

type：跟上级标签一致，如图，本级别标签为COM-EVENT-GRANT-DESIGN。

parameters：参数，本节点的上位机显示的参数都在此配置，如上

COM-EVENT-GRANT-DESIGN包含的参数有“SHORT-NAME”，“PROCESS-DESIGN-REF”，“EVENT-IREF”，其中“EVENT-IREF”又是一个子节点，在childNode中再定义子节点信息。“EVENT-IREF”子节点中又包含了“CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF”，“CONTEXT-R-PORT-PROTOTYPE-REF”，“TARGET-EVENT-REF”三个参数。如下图所示。

****

childNode：子节点，如上“EVENT-IREF”是“COM-EVENT-GRANT-DESIGN”的子节点，

根据需求，子节点可以不断嵌套。

specialRestriction：特殊说明。

restrictionReason：说明原因。

DataType：数据类型，有如下几种选项。

BOOLEAN：bool型。

POSITIVE-INTEGER：整型

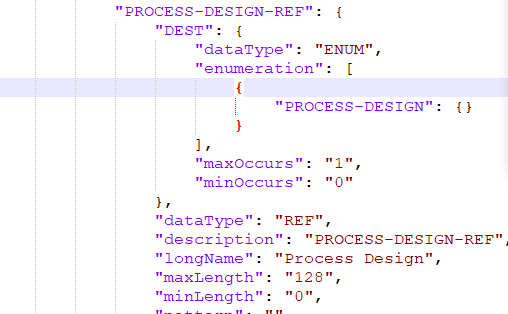
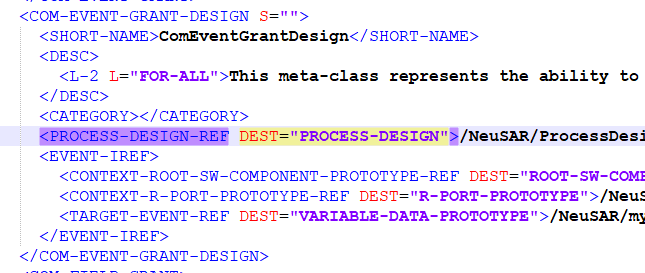
STRING：字符串类型

ENUM：枚举类型，枚举类型后面要接enumeration节点，将枚举陈列其中。

REF：索引类型，当前元素引用其他元素的时候，索引类型跟“DEST”联合使用。

REFS：多引用，当前元素引用其他元素，可以进行多个引用选择的时候，定义REFS，可以参考IamModuleInstantiation节点来实现。

DEST：arxml文件中的参数节点中如果定义了“DEST”，就需要在json文件中定义ENUM枚举类型。如节点<PROCESS-DESIGN-REF DEST="PROCESS-DESIGN">，则需要定义如图的枚举类型。要不然上位机check不过。



**元素顺序属性**：在我们配置上位机check的时候，报错关于元素顺序错误的问题时，是需要在xsd中查看以下元素的属性，如上图中的CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF的属性是属性是sequence，则需要在sequence.json中定义元素顺序属性，要不然check不过。



nonstandard": 我们的arxml是根据AUTOSAR标准制定的，当有非标准的需求时，定义节点中可以添加nonstandard节点，节点属性为"true"，即表示当前节点是非标准定义的节点。

## 4 Python程序编码

在SDC-EOS-aCore\02.Develop\02.SoftwareDevelopment\04.PlatformSourceCode\Branch\_Trunk\_1911\jsonGenerator中，是上位机的python代码，python代码的作用是对上位机上相关的配置信息进行check，根据需求生成代码，生成程序需要的manifest文件和程序自己需要的配置文件。

### 4.1 python的IDE环境配置

(默认已安装python，如果没有，请安装python2.7)

以下以eclipse为利，说明以下工程环境的配置。首先需要在你的python运行环境中安装 tenjin，lxml，pathlib，typing。切记lxml要3.5版本以上的，否则编译不过，程序自然退出且不报错。

pip install tenjin

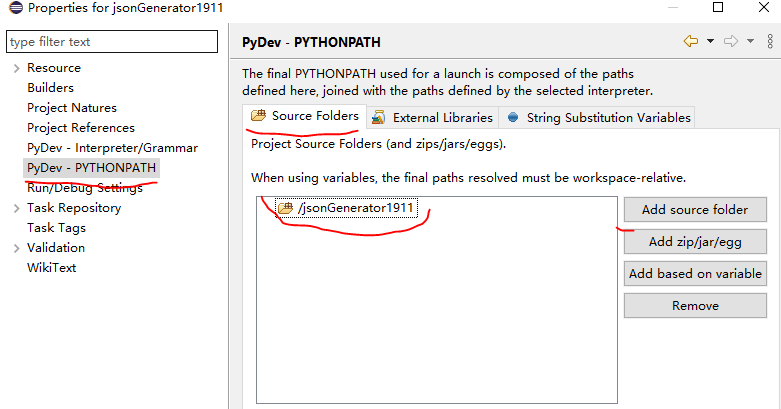
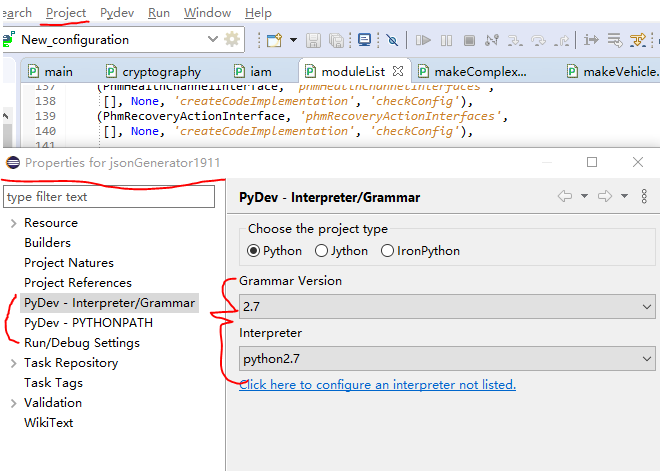
pip install pathlib

pip install typing

pip install lxml

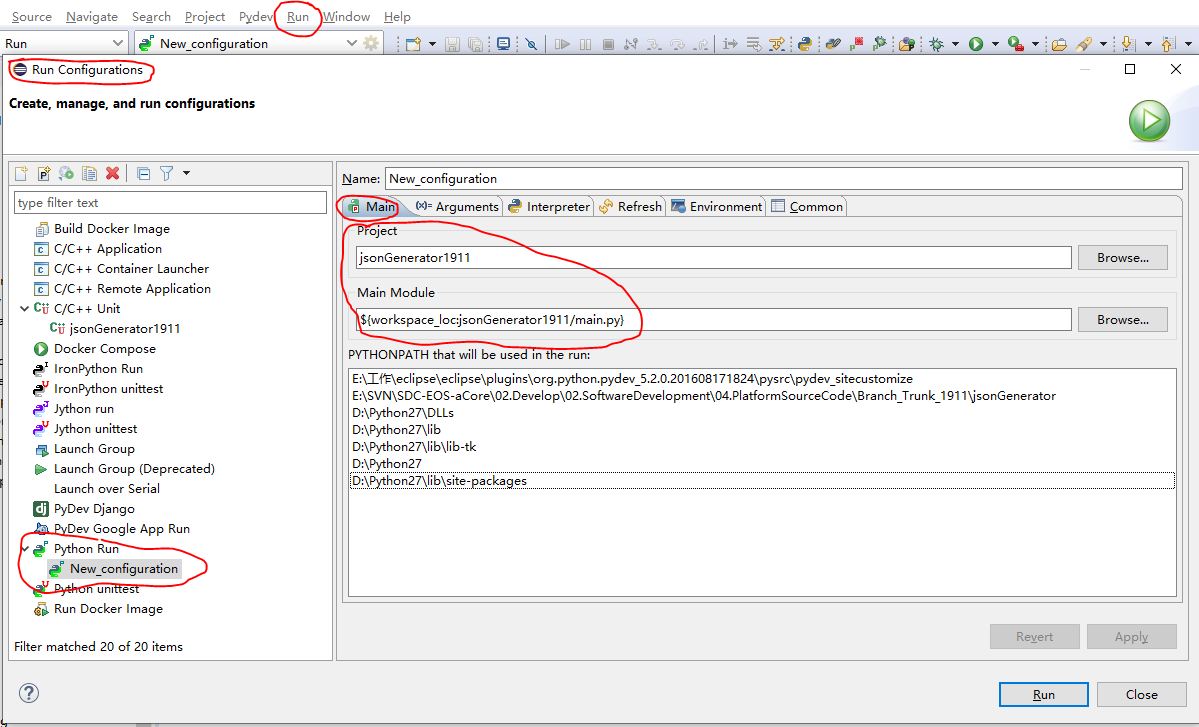
打开eclipse，加载工程jsonGenerator1911。并按照如下步骤设置环境变量。

1 Project->properties:设置Interpreter和PYTHONPATH

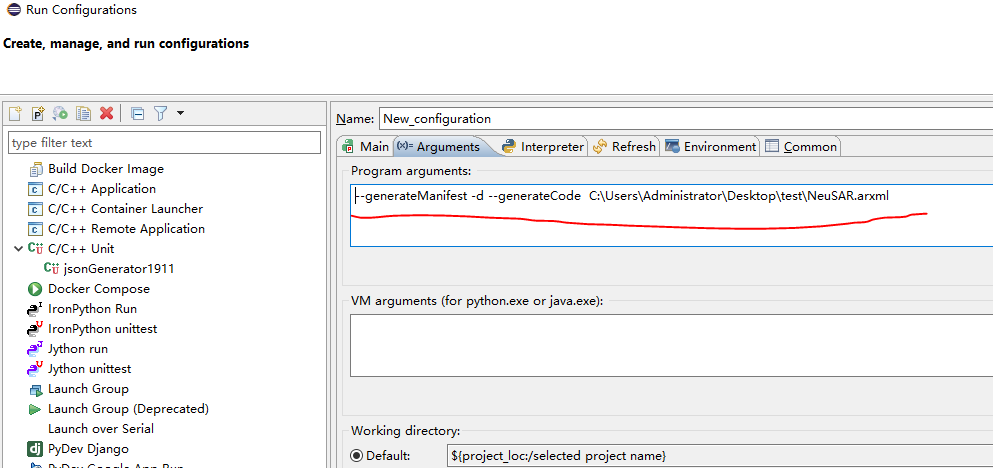


2 Run->Run Configurations->Python Run 中，

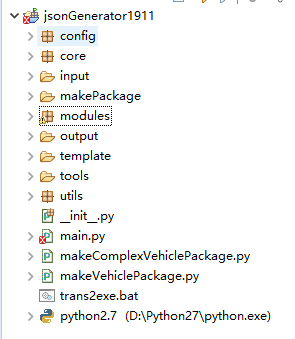
Main：Project里面选择jsonGenerator1911工程。Main Module中选择main.py



Arguments: Program arguments 中填写如下参数：--generateManifest -d --generateCode C:\Users\Administrator\Desktop\test\NeuSAR.arxml（具体自己的NeuSAR.arxml所在的目录路径） -o 输出路径。



### 4.2 代码结构



Input：其中存放这arxml文件，python就是以arxml文件为基础，转换成对应的代码和json。

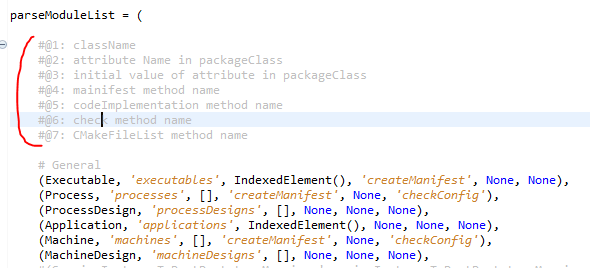
Modules：存放着各个模块的代码。

Output：代码生成的结果输出。

Modules是主要需要修改和添加代码的地方，其中包含各个模块的python包和moduleList.py。下面我们以新增一个IAM模块为例来说明使用。

1. 首先修改moduleList.py，在moduleList.py中的parseModelList 中添加对应模块的类以及其中的方法，参考前面的注释和其他模块的添加方式，根据自己的需求，填写。

（类名成，类包中的属性名称，属性在类包中的个数，manifest文件的创建方法名称，代码执行方法名称，check方法名称）



1. 在添加完parseModuleList中添加完你的模块后，在Modules中创建python包，并在包中实现parseModuleList中定义的各个方法。代码实现的依据是NeuSAR.arxml
2. 在开发的过程中，modulelist中新增的每一个模块，xpath是必须赋值的。并且赋值必须不能跟其他模块的相同，必须是唯一且arxml文件中存在的xpath路径。一下以IAM模块为例

