MVD Checker Guide

Emiel van Strien

M1 Project, University of Technology

05-01-2015

Contents

1. Introduction

This document is created as a guide for anyone who wants to use the MVD Checker created by C. Zhang. The manual is the result of an M1 project, executed by Emiel van Strien at the University of Technology Eindhoven.

The MVD Checker is a tool for IFC validation. The MVD checker is developed based on the open mvdXML standard so it is easily accessed, used and extended by end-users. It can check multiple IFC-files on multiple rules. The rules are defined in XML according to the mvdXML schema. The tool produces one BCF report for each error. These BCF-files can be opened in Solibri to see reports and camera views on the specific errors.

The second chapter tells you which software you need to install and which libraries the MVD Checker uses and how you should add them.
In the third chapter is shown how you should link the MVD Checker to your own data files, how the Checker is used, how you can check multiple mvdXML-rules on multiple IFC-files and it explains how you should analyze the results.
In the fourth chapter the result of two tests executed with this checker are shown. Twelve rules, based on the RGD BIM norm, have been tested. One test is done on an IFC export of an ArchiCAD file and one on an export of a Revit file. It also shows you, if an error on one of the rules is reported, how you should fix them in ArchiCAD and Revit.
Since this tool will let you use your own written rules, in the fifth chapter is shown how to write your own rules in XML using the ifcDoc tool and examples of newly written rules are given.
Because this guide is the result of an M1 project in which this tool is tested, there have been some improvements and fixes to the MVD Checker. One of these improvements is shown in chapter six.

I hope that this document, combined with the MVD Checker Guide Documents, enables you to use the MVD Checker and create/test your own mvdXML-rules.

Emiel van Strien

2. Installing

2.1 Software

To use the MVD checker there are several programs you need to install. Some are mandatory and some are optional.

Mandatory:

- Eclipse IDE for java developers
- Solibri Model Checker v9.1
- Notepad ++
- ifcDoc 6.0
  Can be downloaded at: http://www.buildingsmart-tech.org/specifications/specification-tools/IFCDoc-tool/IFCDoc-download-page
- BIMserver (1.3.1-FINAL-2014-07-21)
  Can be downloaded at: https://github.com/opensourceBIM/BIMserver/releases

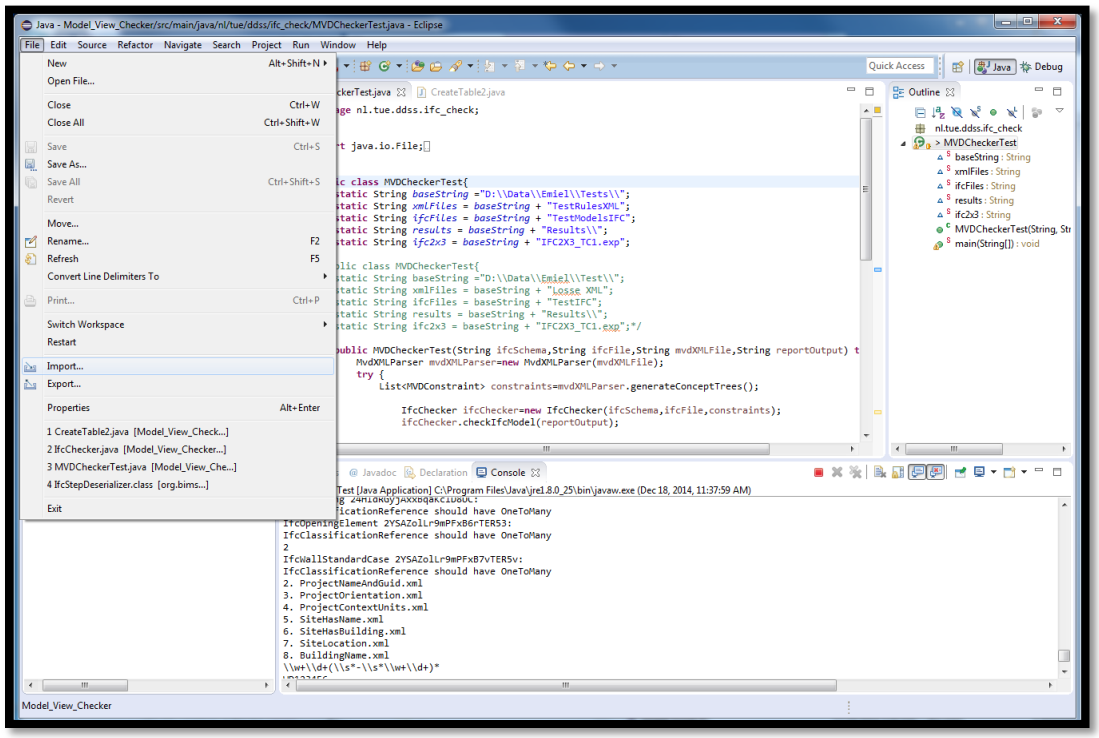
Optional:

- Revit
- ArchiCAD

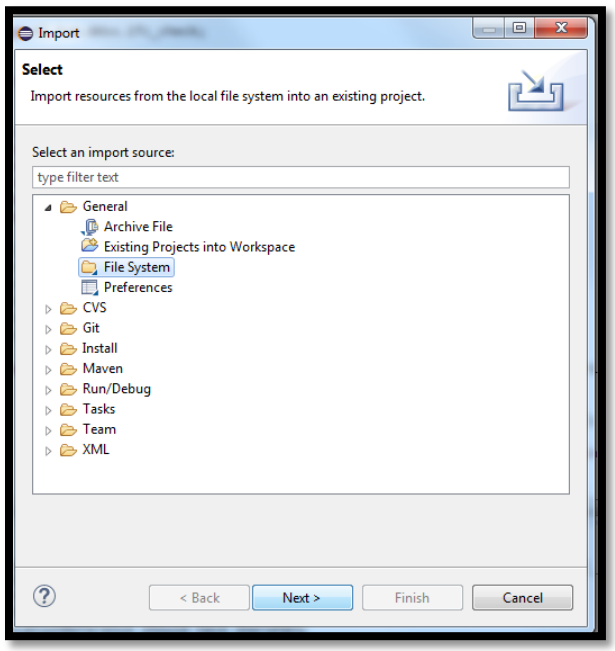
2.2 Libraries

2.2.1 Adding libraries to Eclipse

First you have to load the MVD Checker into Eclipse by going to *File→Import…*



Click *File System* In the *General* tab and then press *Next>*



Browse to the *Model_View_Checker* folder in the *MVD Checker Guide Documents* folder and load this.

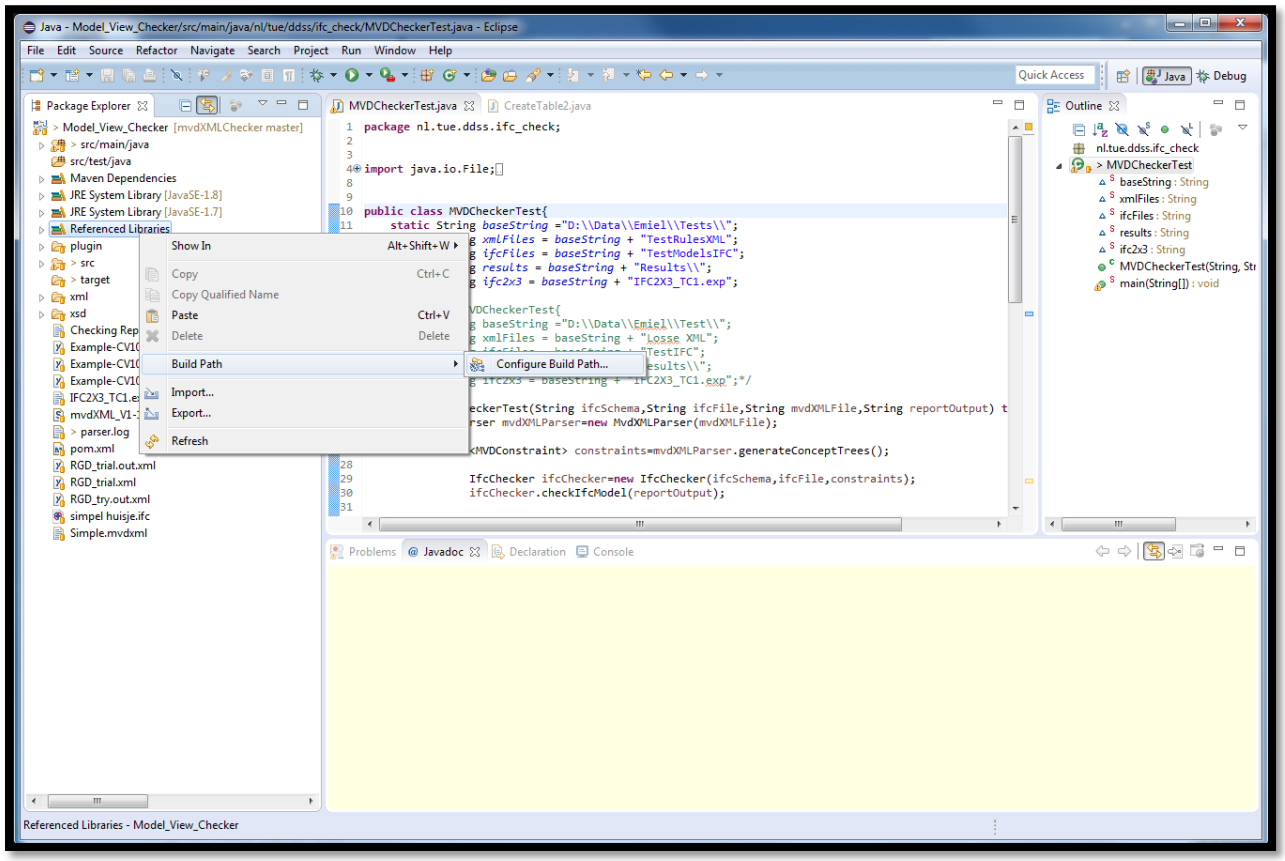
After loading the MVD Checker into Eclipse you need to add multiple libraries to the MVD Checker. These libraries can all be found in the *MVD Checker Guide Documents folder* or can be downloaded from the internet. These are those libraries:

Antlr 3.4 library
Can be downloaded at: https://www.nuget.org/packages/Antlr/3.4.1.9004-pre

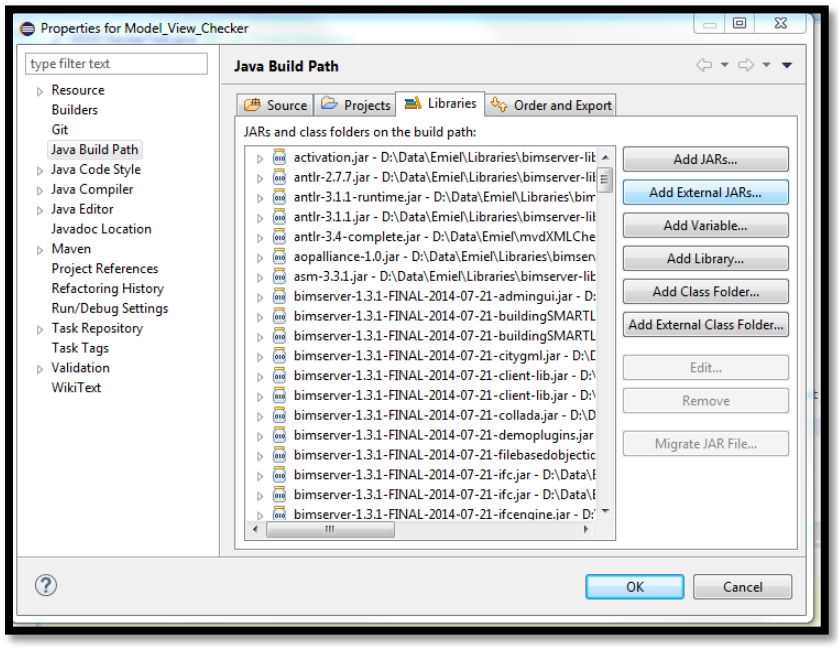
BIMserver library (1.3.1-FINAL-2014-07-21)
Can be downloaded at: https://github.com/opensourceBIM/BIMserver/releases

Vecmath library
Can be downloaded at: http://www.java2s.com/Code/Jar/v/Downloadvecmath151jar.htm

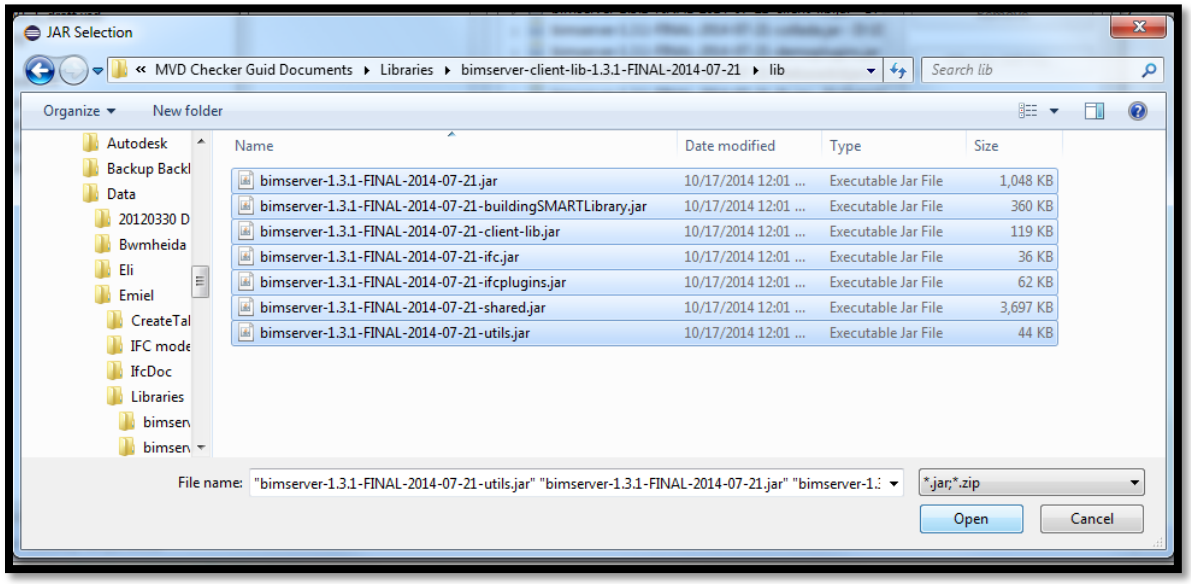
After downloading or copying the libraries, link them to the checker by right clicking on the *Referenced Libraries tab →Build Path →Configure Build Path..*



Click on *Add External JARs..*



Then add al .jar files from the *MVD Checker Guide Documents//Libraries folder*

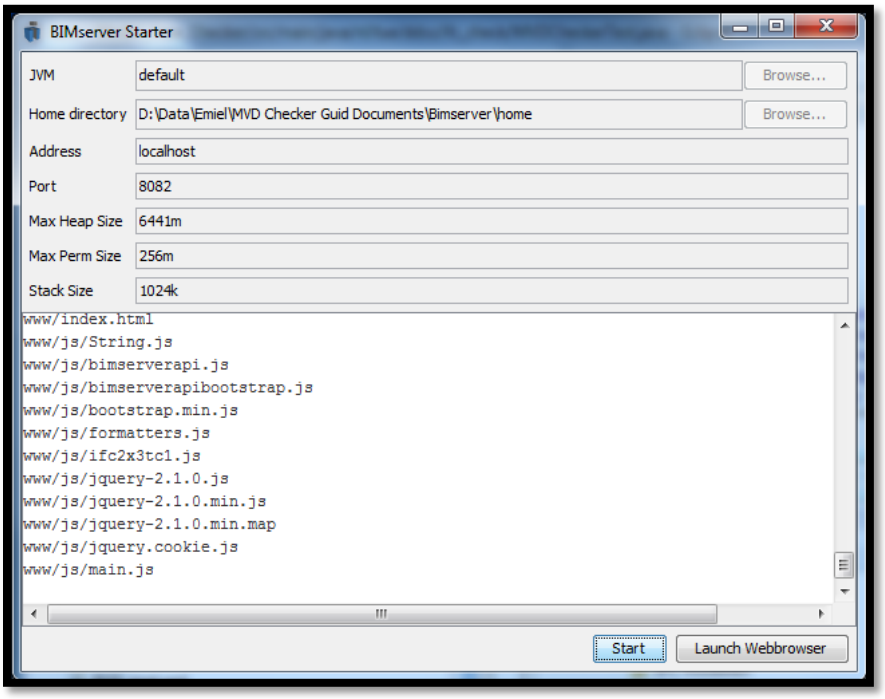


When all libraries are added close the windows until you return to the MVD Checker

2.2.2 Running the BIMserver

Everytime you restart your computer you need to run the BIMserver before you can use the MVD Checker.
To run the BIMserver double click on the *bimserver-1.3.1-FINAL-2014-07-21.jar* file you downloaded or copied from the *MVD Checker Guide Documents* folder. The first time you run the BIMserver you have to click *the Launch Webbrowser* button and create an account.

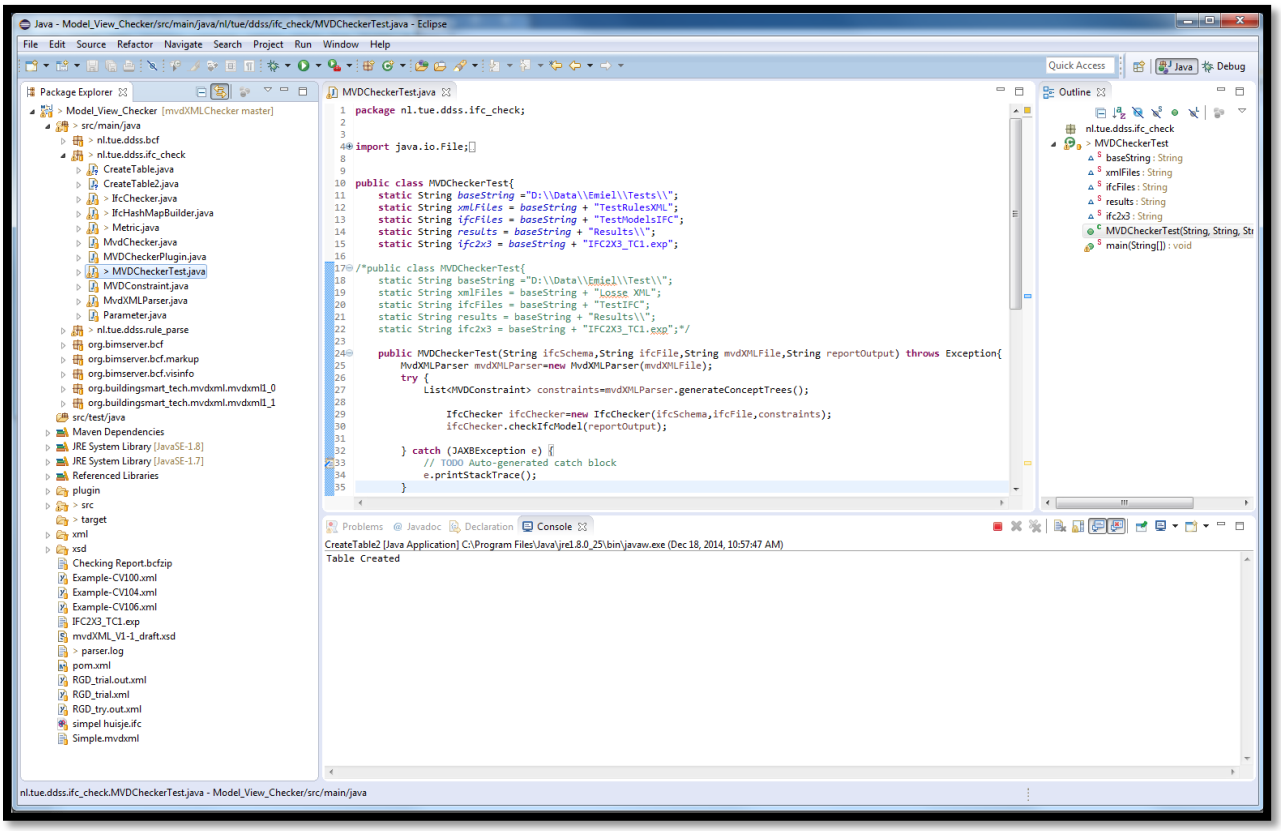
After opening the BIMserver click *start* to connect. (This may take some time)



3. MVD Checker

3.1 Linking the MVD Checker to your data

When everything is installed and the libraries are added the MVD Checker is almost ready to be used. The highlighted script the *>MVDCheckerTest.java* is the actual script which launches the MVD Checker.



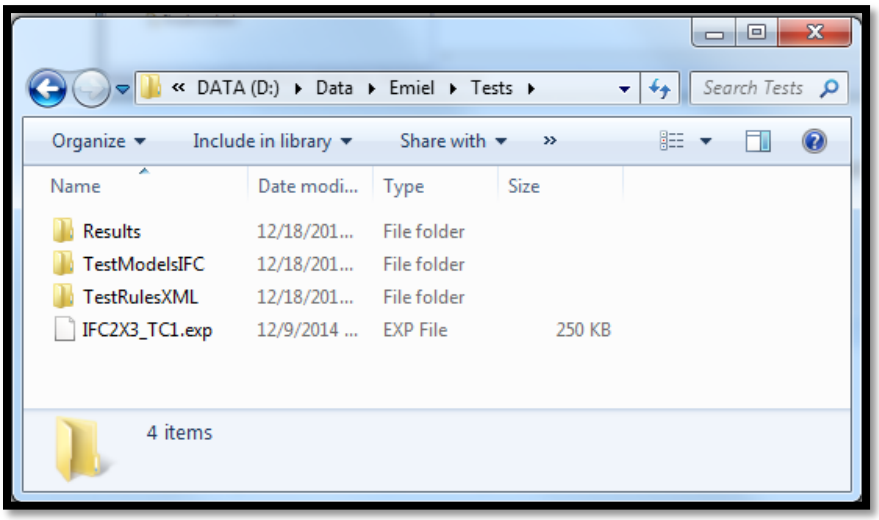
Since you need to link the checker to your own data files you need to change the *BaseString* rule to your own Tests folder.

```
public class MVDCheckerTest{
    static String baseString ="D:\\Data\\Emiel\\Tests\\";
    static String xmlFiles = baseString + "TestRulesXML";
    static String ifcFiles = baseString + "TestModelsIFC";
    static String results = baseString + "Results\\";
    static String ifc2x3 = baseString + "IFC2X3_TC1.exp";
```

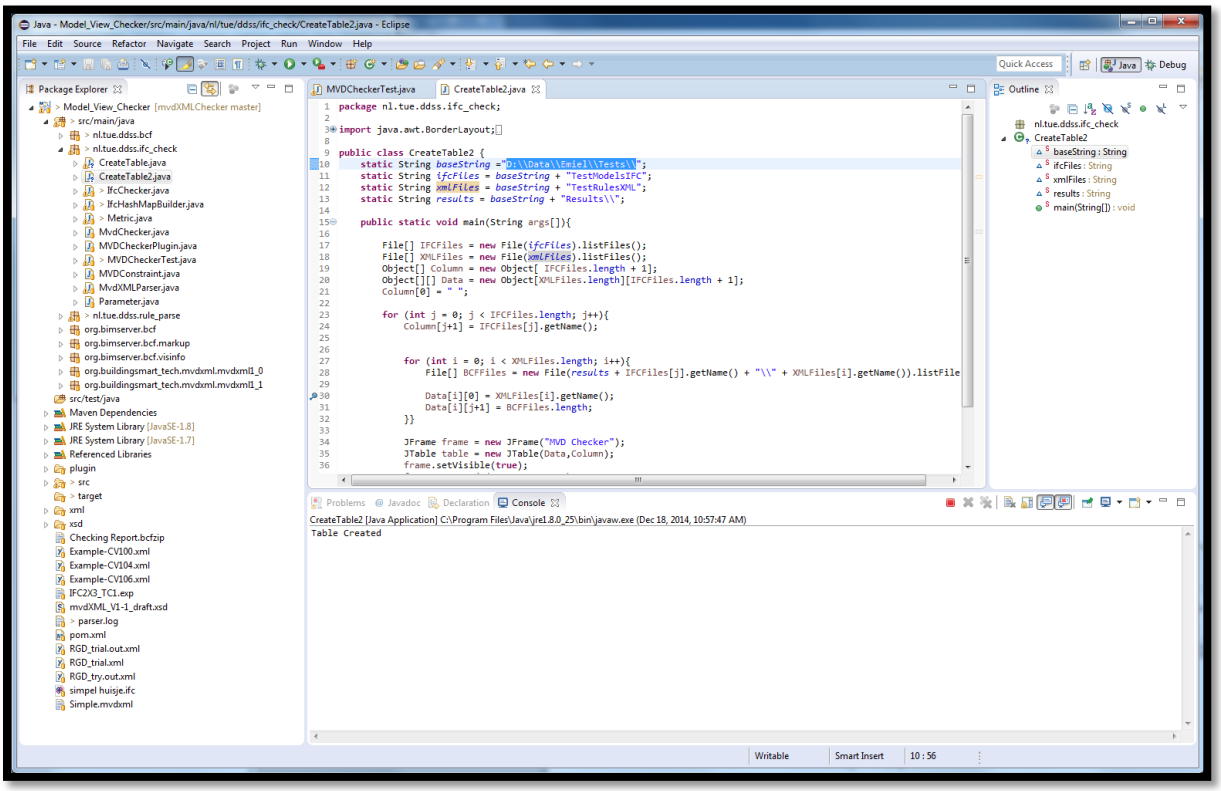
The checker needs 4 different folders/files:

- *xmlFiles:* The rules which are checked
    - Create the folder *TestRulesXML* in the Tests folder
- *ifcFiles:* The IFC-Files that you want to check
    - Create the folder *TestModelsIFC* in the Tests folder
- *results:* A result folder where the results can be placed
    - Create the folder *Results* in the Tests folder
- *ifc2x3:* The basic ifc2x3 file
    - *Add the IFC2X3_TC1*.exp file to the Tests folder

This is what the Tests folder should look like:

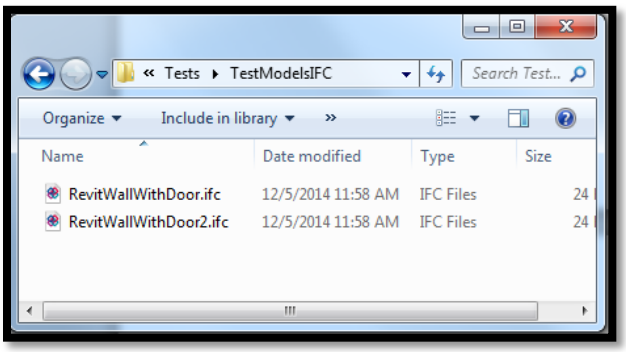


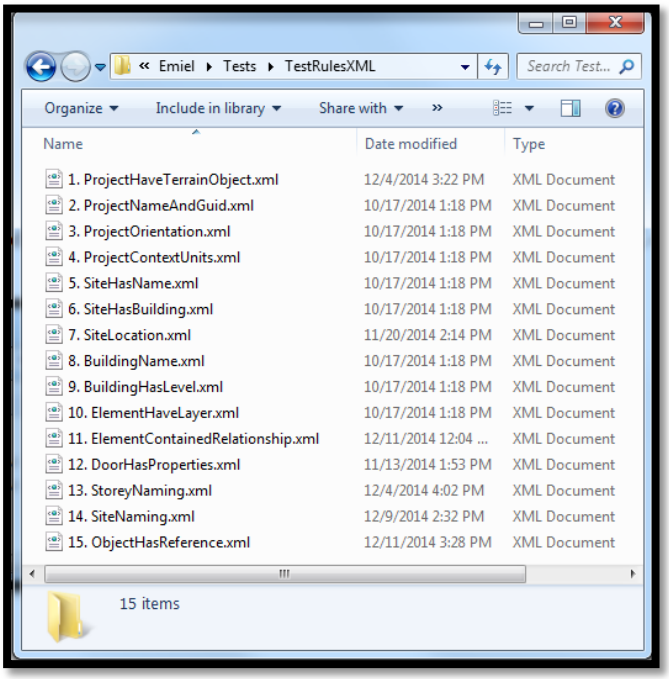
The MVD checker also creates a table to get a quick overview of the errors. This uses the *CreateTable2.java* script. So here you also need to change the *baseString*.



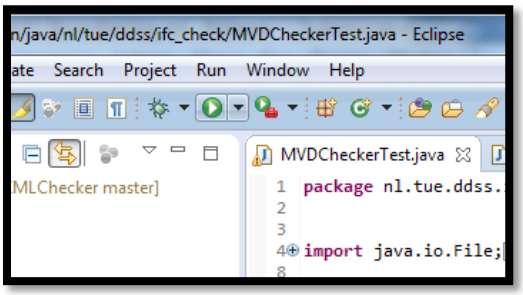
3.2 Using the MVD Checker

To use the MVD Checker you need to put the XML-Rules in the *TestRulesXML* folder and the IFC-models in the *TestModelsIFC* folder.

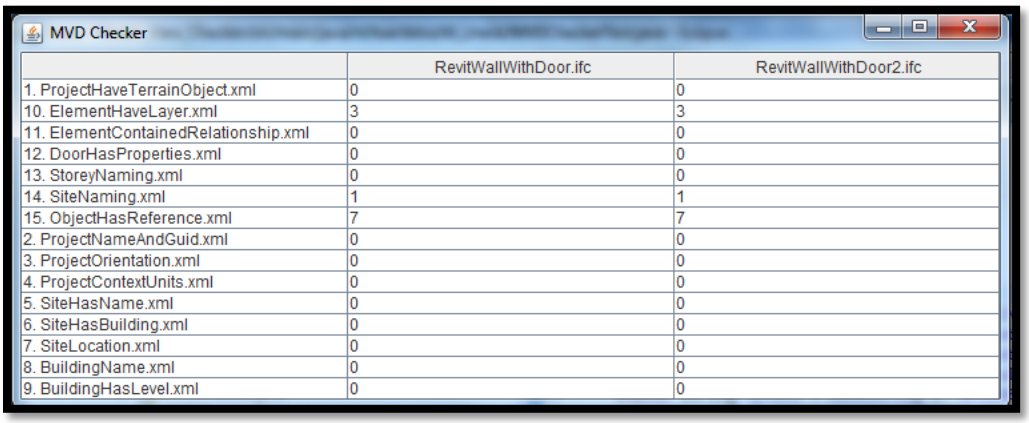




Now you can run the MVD Checker by clicking the *run* button in Eclipse.



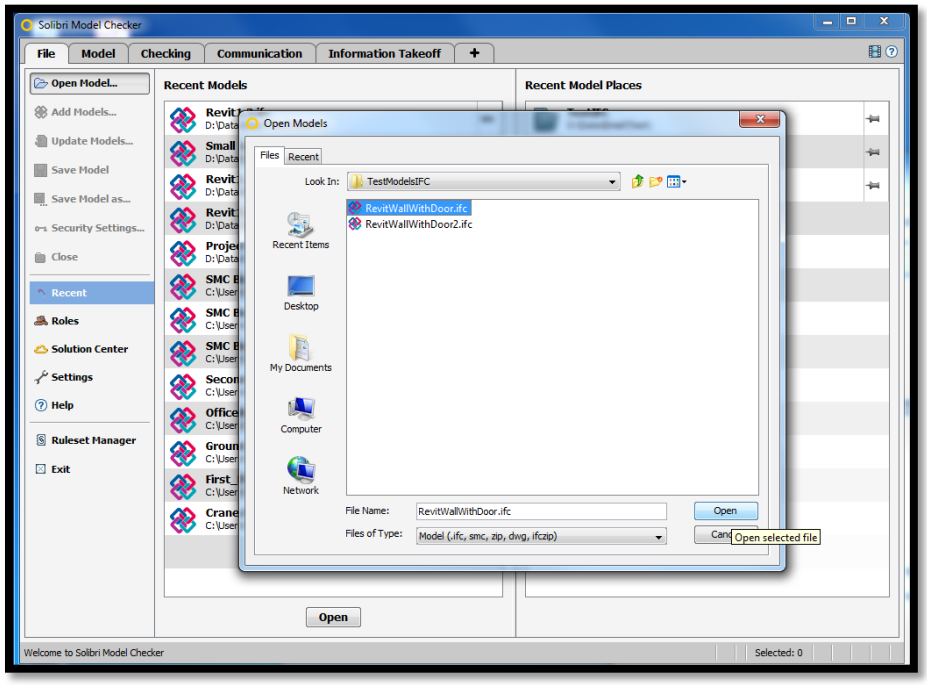
When the MVD Checker is done it creates a table in which you can see on which rules the IFC-Files report errors and how many.


MVD Checker

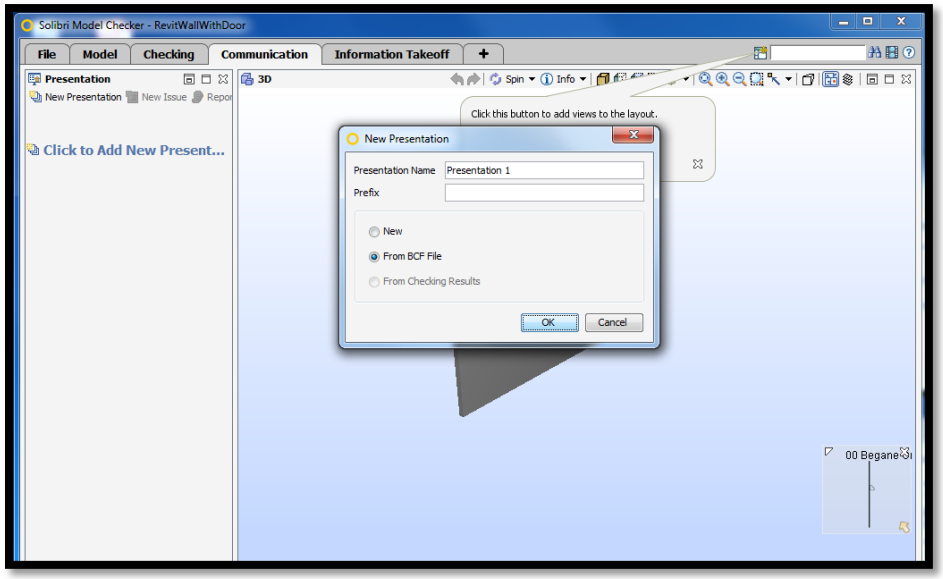
	RevitWallWithDoor.ifc	RevitWallWithDoor2.ifc
1. ProjectHaveTerrainObject.xml	0	0
10. ElementHaveLayer.xml	3	3
11. ElementContainedRelationship.xml	0	0
12. DoorHasProperties.xml	0	0
13. StoreyNaming.xml	0	0
14. SiteNaming.xml	1	1
15. ObjectHasReference.xml	7	7
2. ProjectNameAndGuid.xml	0	0
3. ProjectOrientation.xml	0	0
4. ProjectContextUnits.xml	0	0
5. SiteHasName.xml	0	0
6. SiteHasBuilding.xml	0	0
7. SiteLocation.xml	0	0
8. BuildingName.xml	0	0
9. BuildingHasLevel.xml	0	0

3.3 Viewing the BCF-files

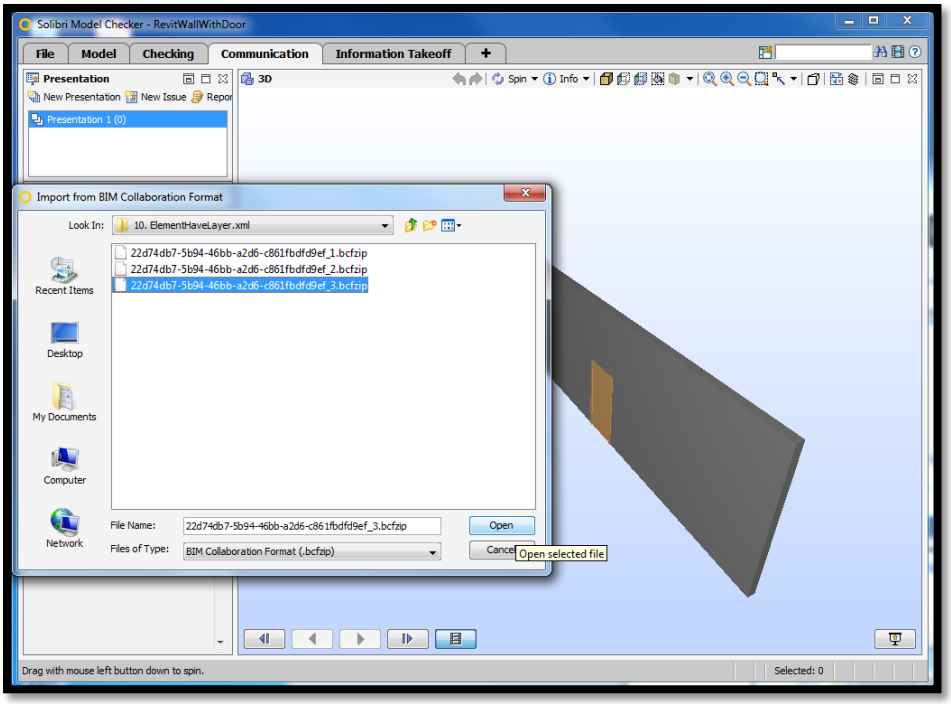
The created table gives you a quick overview of the created errors (one BCF-file is created for each error in the result folder). Currently it works like this but in the future the checker should report multiple similar errors in one BCF-file which makes it much easier to view the the errors in Soliobri. To see more specific details of the errors you can open the created BCF files in Solibri Model Checker. First open the IFC-File.



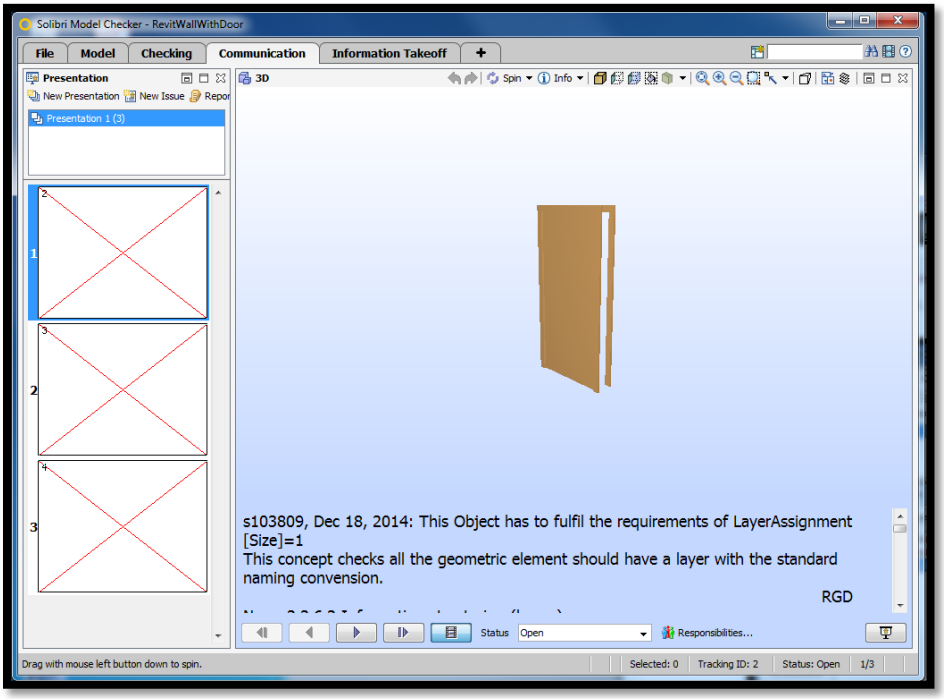
Go to the *Communication* tab and *Click to Add New Presentation*. Here you can select a New Presentation From BCF File.



Navigate to the *Results* folder and open the folder named after the IFC-File. This folder contains folders named after the XML-rules and in there you can choose the BCF file which you want to view. Always choose the last BCF-file since this one contains all the errors.



Since the MVD Checker reported 3 errors on this rule (10. ElementHaveLayer), 3 different camera views are created of 3 different elements. By clicking on these views you go to the specific view of this element with a report of the error. When no specific camera view can be created it creates a general overview of the complete project.



4. Tests

4.1 Revit Test

4.1.1 Information

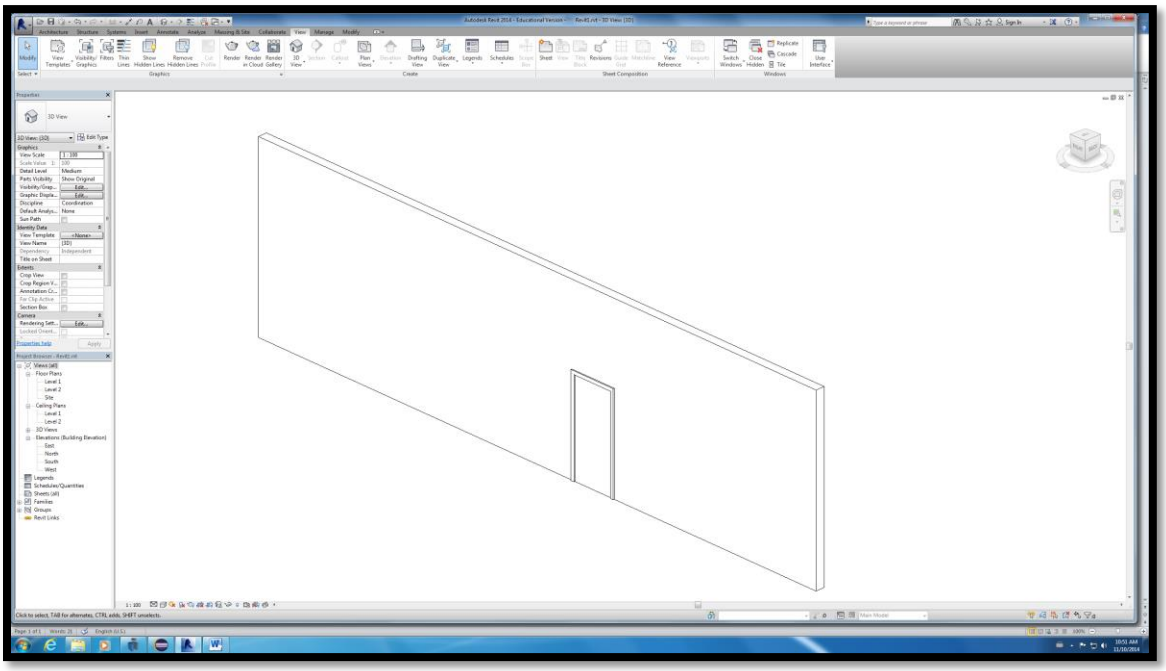
Name model: Revit1

Date: 10-11-2014

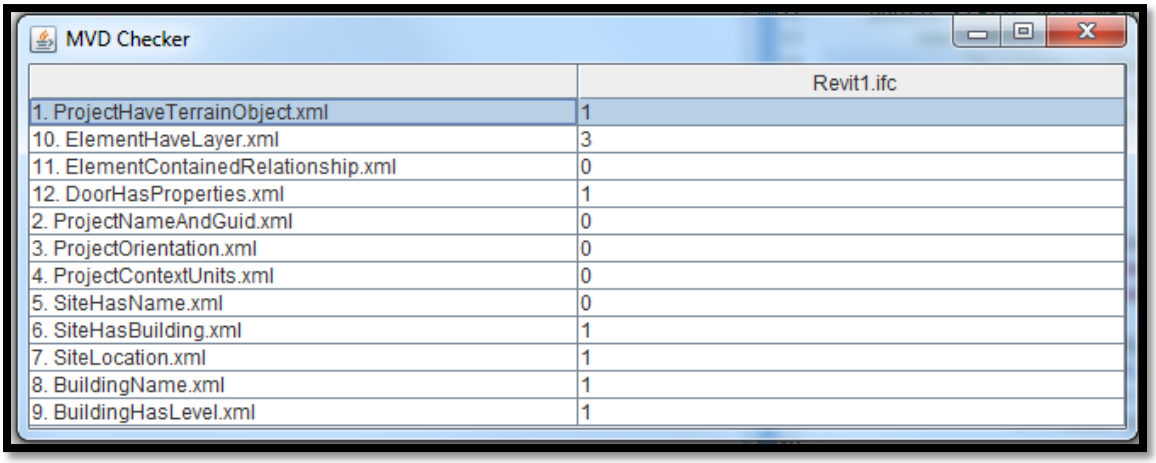
Software: Autodesk Revit 2014

Model: Simple wall with door

XML rules: 12



The model



	Revit1.ifc
1. ProjectHaveTerrainObject.xml	1
10. ElementHaveLayer.xml	3
11. ElementContainedRelationship.xml	0
12. DoorHasProperties.xml	1
2. ProjectNameAndGuid.xml	0
3. ProjectOrientation.xml	0
4. ProjectContextUnits.xml	0
5. SiteHasName.xml	0
6. SiteHasBuilding.xml	1
7. SiteLocation.xml	1
8. BuildingName.xml	1
9. BuildingHasLevel.xml	1

Errors created on the rules

4.1.2 XML-Rules

4.1.2.1 ProjectHaveTerrainObject

Information

This rule checks if the ifcProject has at least one TerrainObject (ifcSite). One error was reported. But when manually checking the IFC-File, the ifcProject had one TerreinObject so it shouldn't give an error.

4.1.2.2. ProjectNameAndGuid

Information

This rule checks if the project has a Name and Guid. No error was reported and the IFC-File was manually checked to confirm.

4.1.2.3. ProjectOrientation

Information

This rule checks if the project has an orientation. No error was reported and the IFC-File was manually checked to confirm.

4.1.2.4. ProjectContextUnits

Information

This rule checks if the project units are given in meters. No error was reported and the IFC-File was manually checked to confirm.

4.1.2.5. SiteName

Information

This rule checks if the project has a Name and Guide. No error was reported and the IFC-File was manually checked to confirm.

4.1.2.6. SiteHasBuilding

Information

This rule checks if the ifcSite has at least one ifcBuilding. One error was reported. But when manually checking the IFC-File, the ifcSite has one ifcBuilding so it shouldn't give an error.

4.1.2.7. SiteLocation

Information

This rule checks the global location of the terrain. It should have a refLatitude, refLongtitude and a refElevation. The checker doesn't create a BCF file for this rule, but does report an error in the console. This means there is an error, but the MVD Checker is impossible to create a BCF-file.

Modifications

When manually checking the IFC-File there are several values for refLatitude, refLongtitude and a refElevation.

```
#534= IFCSTYLEDITEM(#526,(#532),$);
#537= IFCSHAPEREPRESENTATION(#73,'Facetation','SurfaceModel',(#526));
#539= IFCPRODUCTDEFINITIONSHAPE($,$,(#537));
#541= IFCCARTESIANPOINT((0.00130975851057826,-0.0212487824024294,0.));
#543= IFCAXIS2PLACEMENT3D(#541,$,$);
#544= IFCLOCALPLACEMENT($,#543);
#545= IFCSITE('1IVNg7Ukz7fAiV0AVlaCXO',#41,'Surface:197980',$,'',#544,#539,$,.ELEMENT.,(51,26,52,367248),(5,29,9,419975),0.,$,$);
#550= IFCPROPERTYSINGLEVALUE('Projected Area',$,IFCAREAMEASURE(360.935902021594),$);
#551= IFCPROPERTYSINGLEVALUE('Surface Area',$,IFCAREAMEASURE(360.935902021594),$);
#552= IFCPROPERTYSET('0RV13wWYPCYfnkTrg2We4i',#41,'Phasing',$,(#156));
#554= IFCRELDEFINESBYPROPERTIES('38Abf8Sw94lvkz$SUGMqoc',#41,$,$,(#545),#552);
#558= IFCPROPERTYSET('0qoz$WkfXFJAj30ivBbFcr',#41,'Dimensions',$,(#550,#551));
```

But this attribute rule in the XML-file expects a value with the cardinality of one.

```
        <ConceptTemplate uuid="568bf132-de9f-4194-9443-51d9509a9d2b"
            name="SiteGlobalLocation" applicableSchema="IFC4" applicableEntity="IfcSite">
            <Rules>
                <AttributeRule AttributeName="RefLatitude"
                    Cardinality="One" />
                <AttributeRule AttributeName="RefLongitude"
                    Cardinality="One" />
                <AttributeRule AttributeName="RefElevation"
                    Cardinality="One" />
            </Rules>
        </ConceptTemplate>
```

So I changed the attribute values manually into one number like this:

```
#541= IFCCARTESIANPOINT((0.00130975851057826,-0.0212487824024294,0.));
#543= IFCAXIS2PLACEMENT3D(#541,$,$);
#544= IFCLOCALPLACEMENT($,#543);
#545= IFCSITE('1IVNg7Ukz7fAiV0AVlaCXO',#41,'Surface:197980',$,'',#544,#539,$,.ELEMENT.,(51),(5),0.,$,$);
#550= IFCPROPERTYSINGLEVALUE('Projected Area',$,IFCAREAMEASURE(360.935902021594),$);
#551= IFCPROPERTYSINGLEVALUE('Surface Area',$,IFCAREAMEASURE(360.935902021594),$);
```

After doing that it didn't give an error anymore on this rule, but since the value of the attribute RefLatitude etc. is normally a List[3:4], the checker should check if this is a List. But currently the checker doesn't support this feature so this should be added in the checker so it also supports this Cardinality. To make the checker work for now the Cardinality in the xml file should be changed from "One" to "OneToMany".

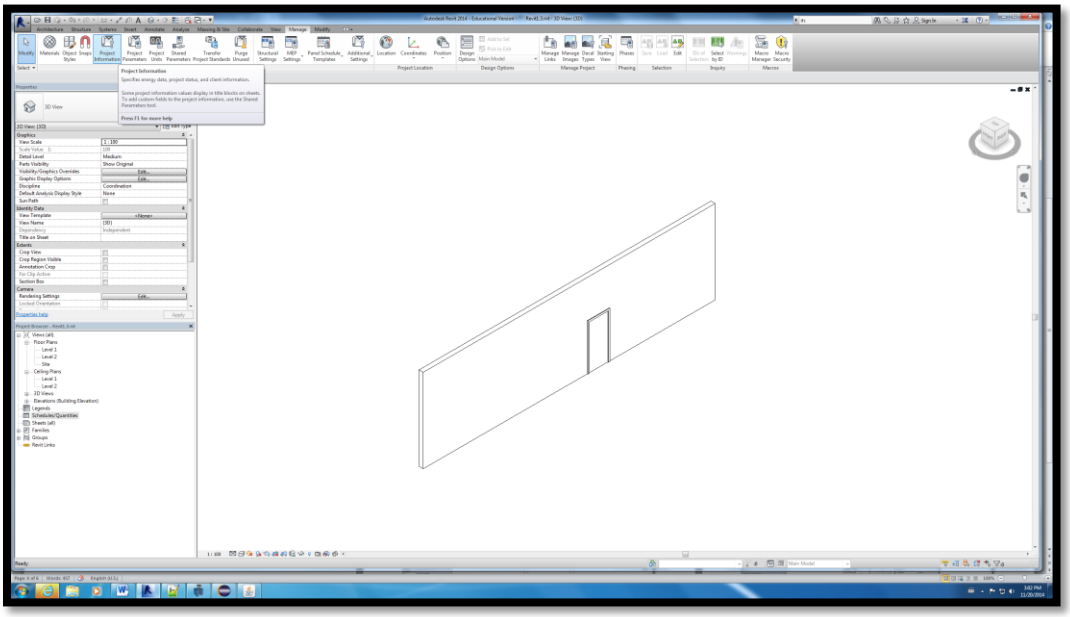
4.1.2.8. BuildingName

Information

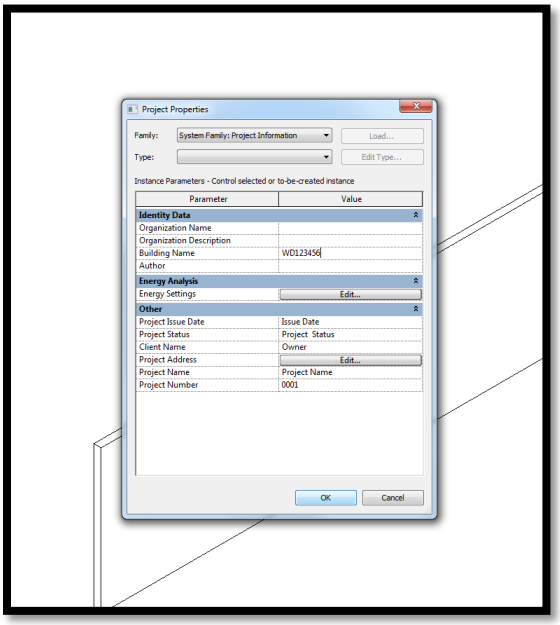
This rule checks the name of building. This name should follow the convention of <Rgd object number>.
RGD Norm 2.2.7.3
Building Attributes:o Name: <Rgd object number> Example: OR123456
If the project consists of multiple Rgd object numbers, all numbers are given, separated by <space>"-"<space>.
Example: OR123456 - OR123457]]>

Modifications

1. In the manage tab click on the *Project Information* button.



2. Change the building name according to the RGD Norm 2.2.7.3. and press OK.



4.1.2.9. BuildingHasLevel

Information

This rule checks if the ifcBuilding has at least one ifcBuildingStorey. One error was reported. But when manually checked, the ifcSite has one ifcBuilding so it shouldn't give an error.

4.1.2.10. ElementHaveLayer

Information

This rule checks if every ifcElement has a layerassignment. This rule gives 3 errors (one for each element in the IFC-File). But when manually checked, the wall and the door element have a layerassignment, so this shouldn't give 3, but one error.

4.1.2.11. ElementContainedRelationship

Information

This rule checks if all elements contain a relationship with either a space or a buildingStorey. No error was reported and the IFC-File was manually checked to confirm

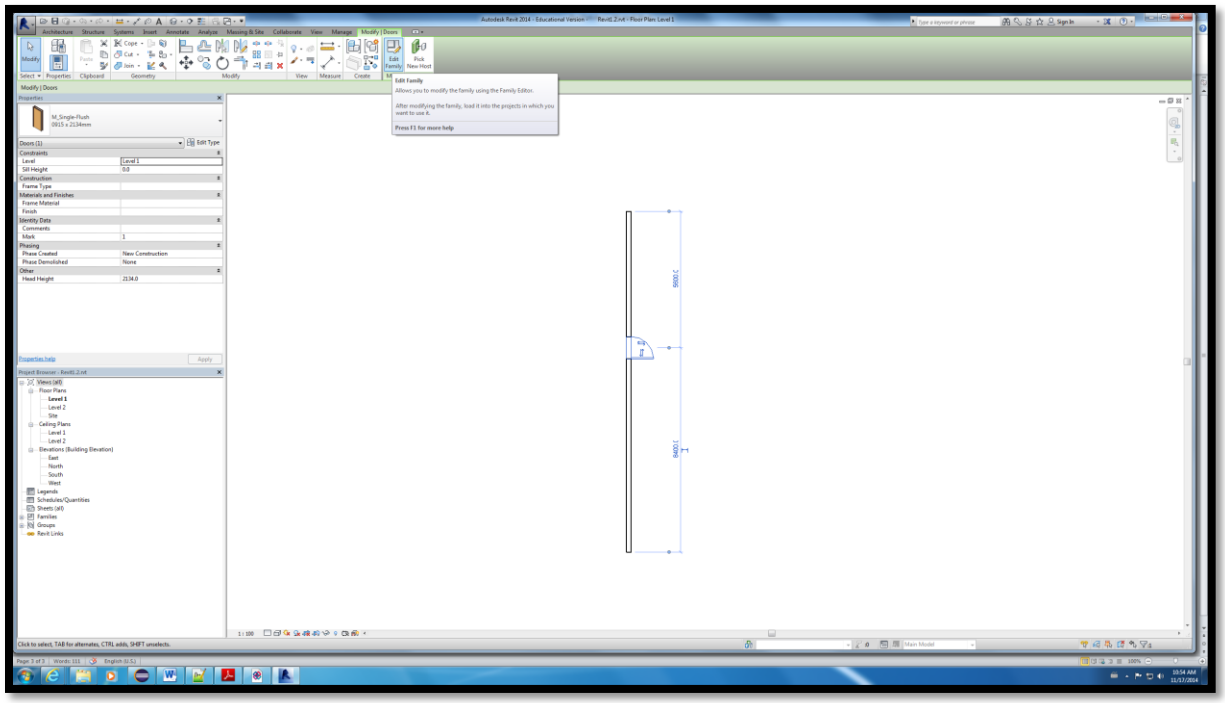
4.1.2.12. DoorHasProperties

Information

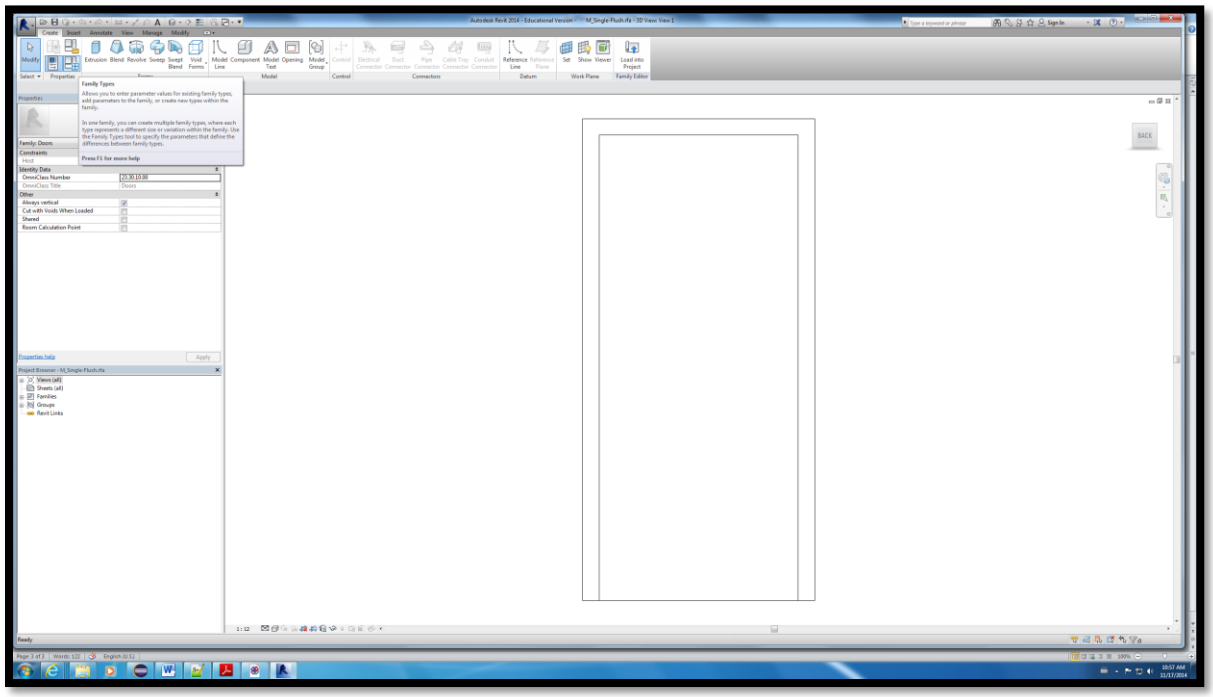
Rule 12 checks if a door has the properties of SelfClosing, FireExit and SmokeStop regardless of the value (True/False). On this rule one error was reported and one BCF-file was created

Modifications (Fix rule 12)

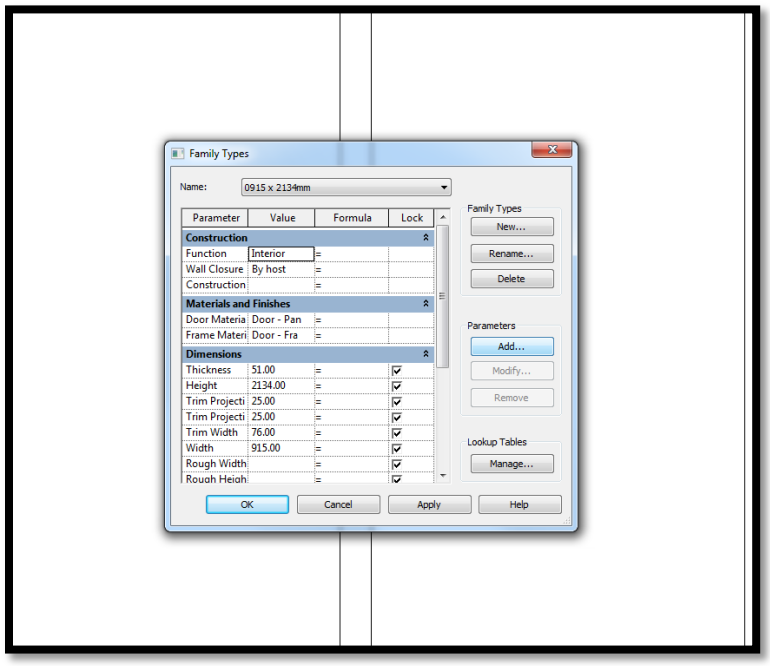
1. Click the door and click the *Edit Family* button.



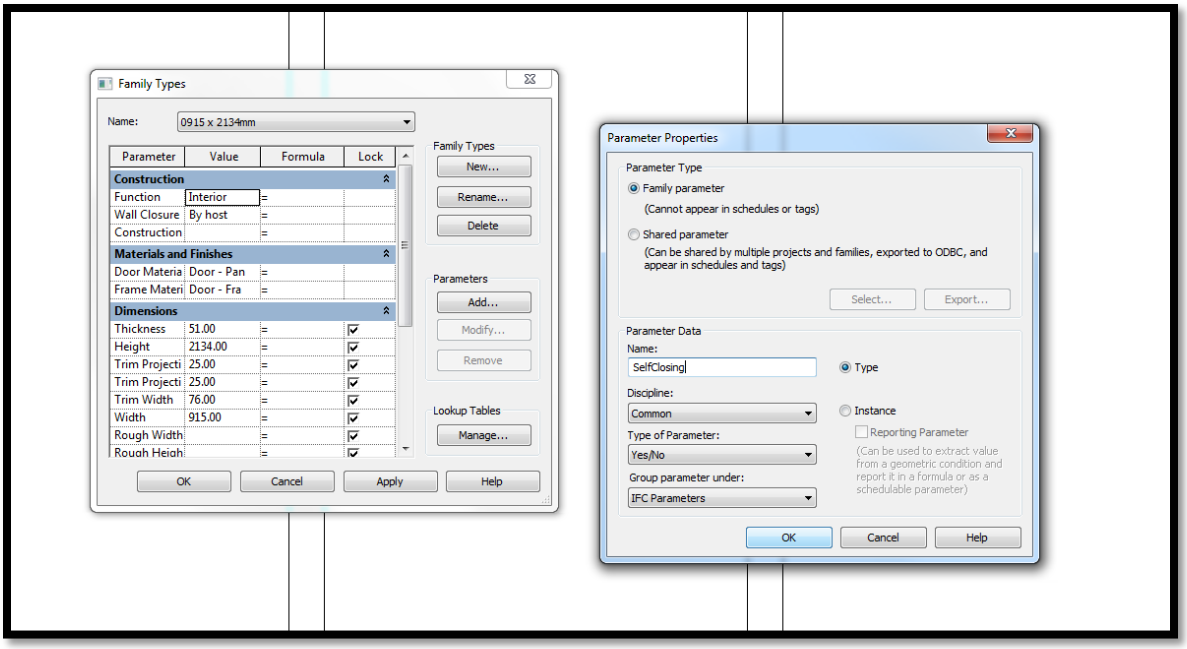
2. Click the *Family Type* button.



3. Add parameters to the door.

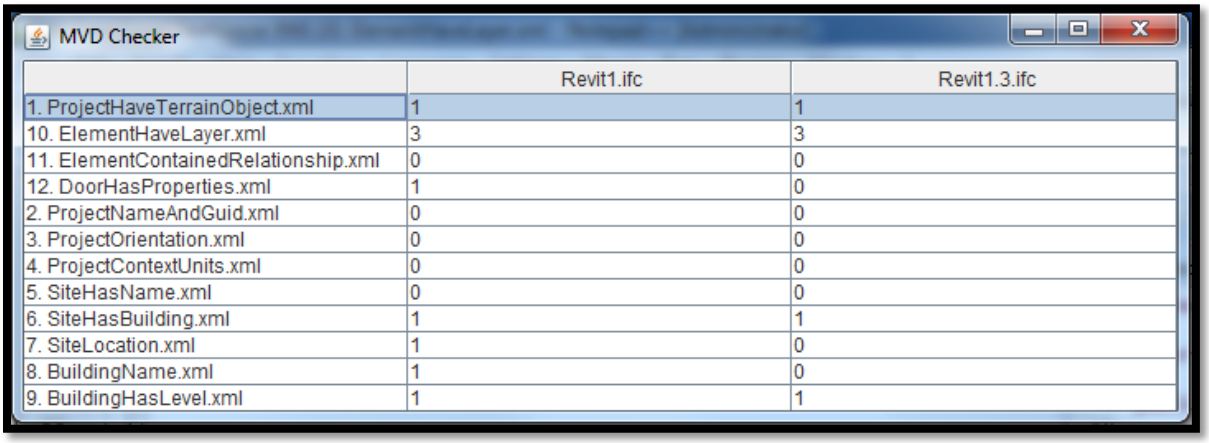


4. Choose the name, discipline, type of parameter and the group of parameters. Then press *OK*.



5. After you pressed *Apply* and *OK,* press the *Load into Project* button.

4.1.3 Result



	Revit1.ifc	Revit1.3.ifc
1. ProjectHaveTerrainObject.xml	1	1
10. ElementHaveLayer.xml	3	3
11. ElementContainedRelationship.xml	0	0
12. DoorHasProperties.xml	1	0
2. ProjectNameAndGuid.xml	0	0
3. ProjectOrientation.xml	0	0
4. ProjectContextUnits.xml	0	0
5. SiteHasName.xml	0	0
6. SiteHasBuilding.xml	1	1
7. SiteLocation.xml	1	0
8. BuildingName.xml	1	0
9. BuildingHasLevel.xml	1	1

Rule 1, 6 and 9 all report an error. But when manually checked in the IFC-file, these rules should give errors. Since all of these rules use the same concept template “Aggregation” there must be something wrong with this rule or with MVD Checker. In chapter 6 this problem is solved.
Rule 10. ElementHaveLayer also doesn’t report correct errors. This rule isn’t correctly supported in the MVD Checker. But we we’re unable to fix this since the BIMserver library does not return the value for the attribute of IfcPresentationLayerAssignment.
Rule 7 didn’t create a BCF-file. But did report an error in the console. It didn’t create a BCF file because not for all entities a specific camera view could be made because this is not supported by the Collada serializer (https://code.google.com/p/bimserver/wiki/ColladaSerializer). The MVD Checker was rewritten so it would, if no specific camera view could be generated, use a overview camera of the entire project.

4.2 ArchiCAD Test

4.2.1 Information

Name model: ArchiCad1

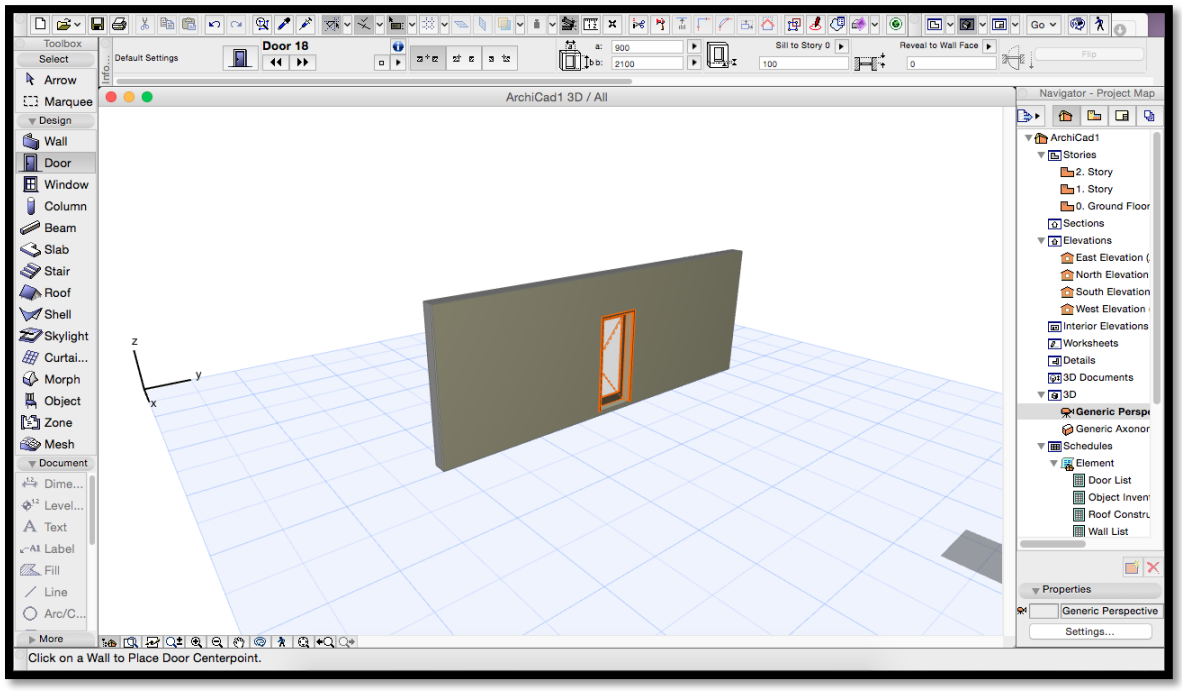
Date: 09-12-2014

Software: ArchiCAD 18

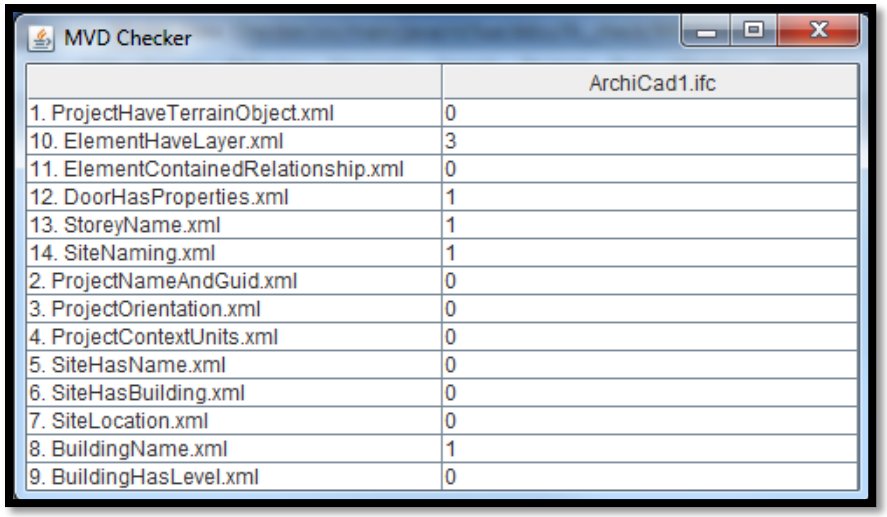
Model: Simple wall with door

XML rules: 14 (12 + 2 additional developed rules)

Extra: This test is based on the revised version of this tool according to identified errors in the first RevitTest.



The ArchiCad model



	ArchiCad1.ifc
1. ProjectHaveTerrainObject.xml	0
10. ElementHaveLayer.xml	3
11. ElementContainedRelationship.xml	0
12. DoorHasProperties.xml	1
13. StoreyName.xml	1
14. SiteNaming.xml	1
2. ProjectNameAndGuid.xml	0
3. ProjectOrientation.xml	0
4. ProjectContextUnits.xml	0
5. SiteHasName.xml	0
6. SiteHasBuilding.xml	0
7. SiteLocation.xml	0
8. BuildingName.xml	1
9. BuildingHasLevel.xml	0

Errors created on the rules

4.2.2 XML Rules

4.2.2.1. ProjectHaveTerrainObject

Information

This rule checks if the ifcProject has a one TerrainObject (ifcSite). No error was reported and when manually checking the IFC-File this was confirmed.

4.2.2.2. ProjectNameAndGuid

Information

This rule checks if the project has a Name and Guid. No error was reported and when manually checking the IFC-File this was confirmed.

4.2.2.3. ProjectOrientation

Information

This rule checks if the project has an orientation. No error was reported and when manually checking the IFC-File this was confirmed.

4.2.2.4. ProjectContextUnits

Information

This rule checks if the project units are given in meters. No error was reported and when manually checking the IFC-File this was confirmed.

4.2.2.5. SiteName

Information

This rule checks if the project has a Name and Guid. No error was reported and when manually checking the IFC-File this was confirmed.

4.2.2.6. SiteHasBuilding

Information

This rule checks if the ifcSite has at least one ifcBuilding. No error was reported and when manually checking the IFC-File this was confirmed.

4.2.2.7. SiteLocation

Information

This rule checks the global location of the terrain. It should have a refLatitude, refLongtitude and a refElevation. No error was reported and when manually checking the IFC-File this was confirmed.

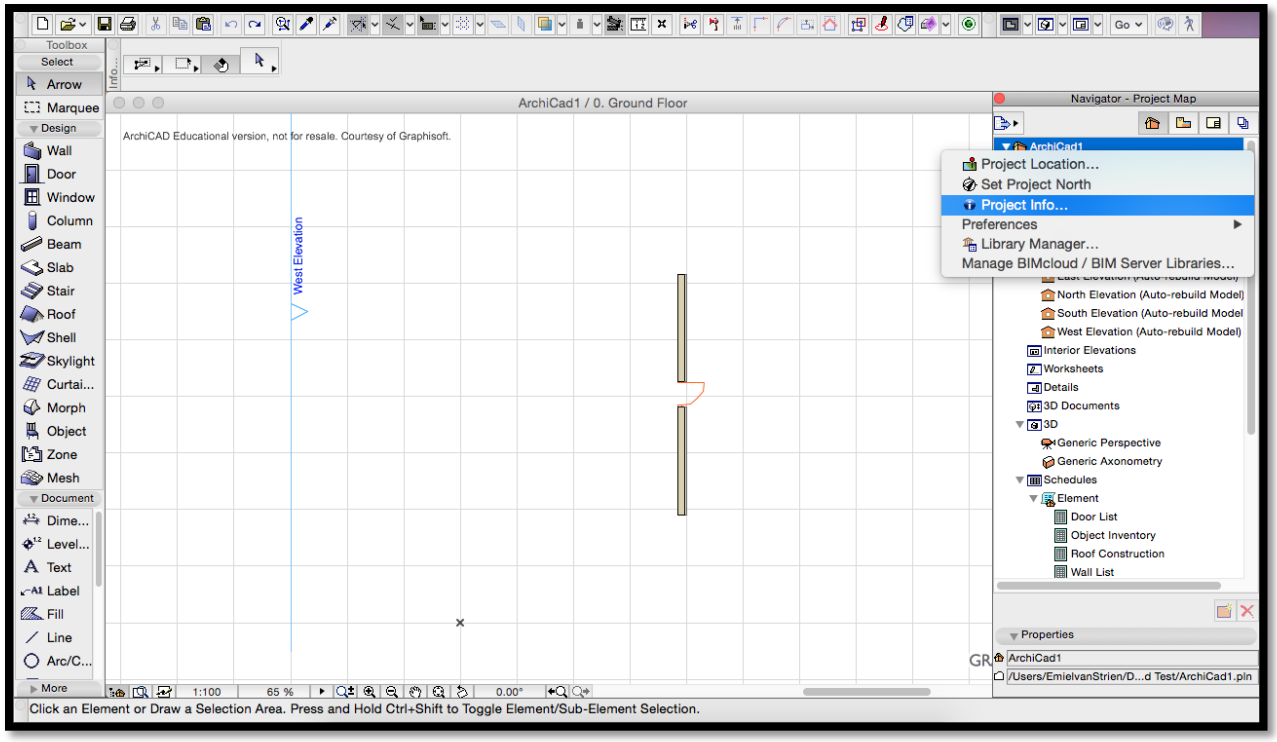
4.2.2.8. BuildingName

Information

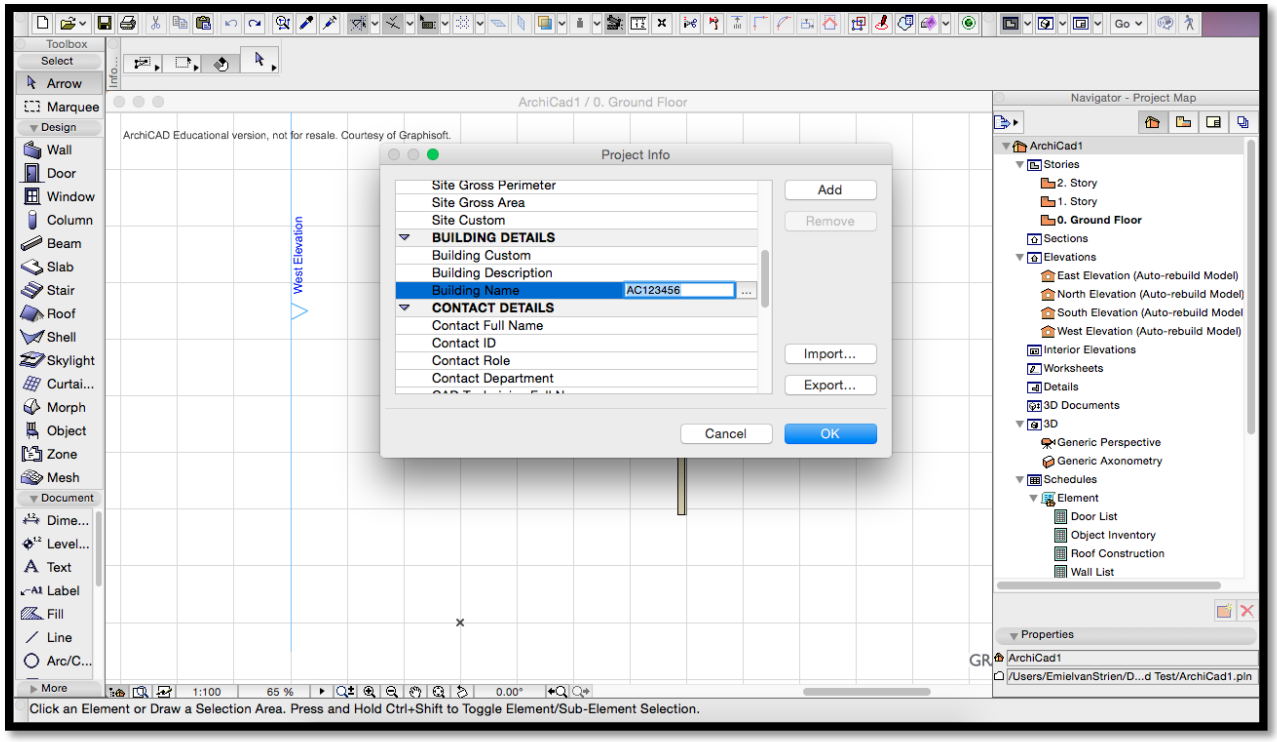
This rule checks the name of building. This name should follow the convention of <Rgd object number>
RGD Norm 2.2.7.3
Building Attributes: Name: <Rgd object number> Example: OR123456
If the project consists of multiple Rgd object numbers, all numbers are given, separated by <space>"-"<space>.
Example: OR123456 - OR123457]]>

The checker reported one error on this rule.

Modifications

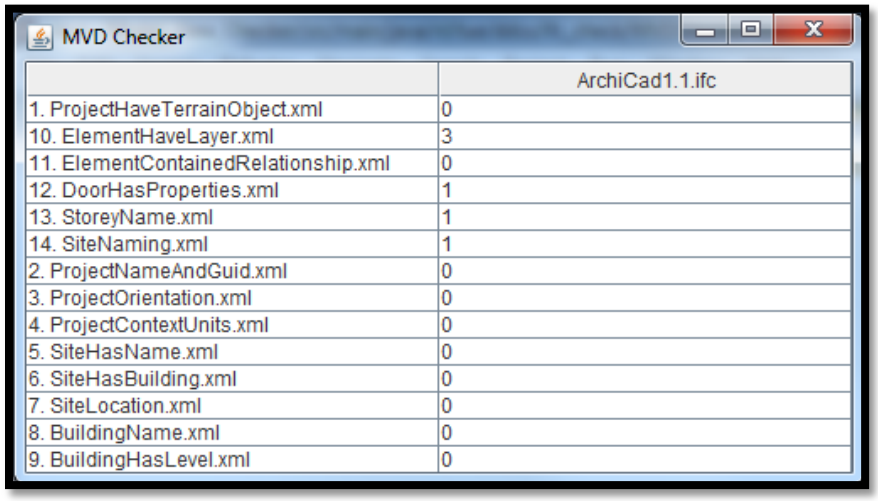


In the navigator on the right side right-click the *project button*. Then click on *Project Info…*



There you can change the name of the Building according to the Rule.

Result



	ArchiCad1.1.ifc
1. ProjectHaveTerrainObject.xml	0
10. ElementHaveLayer.xml	3
11. ElementContainedRelationship.xml	0
12. DoorHasProperties.xml	1
13. StoreyName.xml	1
14. SiteNaming.xml	1
2. ProjectNameAndGuid.xml	0
3. ProjectOrientation.xml	0
4. ProjectContextUnits.xml	0
5. SiteHasName.xml	0
6. SiteHasBuilding.xml	0
7. SiteLocation.xml	0
8. BuildingName.xml	0
9. BuildingHasLevel.xml	0

4.2.2.9. BuildingHasLevel

Information

This rule checks if the ifcBuilding has at least one ifcBuildingStorey. No error was reported and by manually checking the IFC-File this was confirmed.

4.2.2.10. ElementHaveLayer

Information

This rule checks if every ifcElement has a layerassignment. This rule gives 3 errors (one for each element in the IFC-File). But when manually checked, the wall and the door element have a layerassignment, so this shouldn't give 3, but one error.

4.2.2.11. ElementContainedRelationship

Information

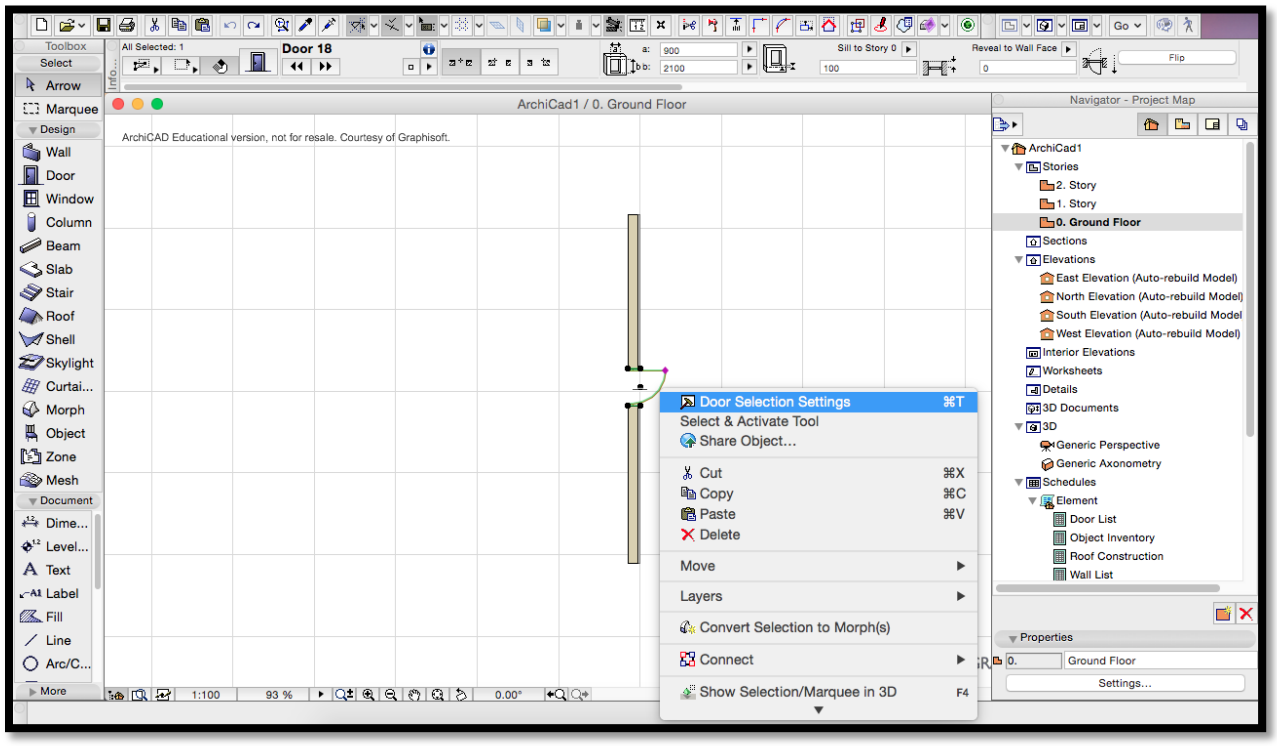
This rule checks if all elements contain a relationship with either a space or a buildingStorey. No error was reported and when manually checking the IFC-File this was confirmed.

4.2.2.12. DoorHasProperties

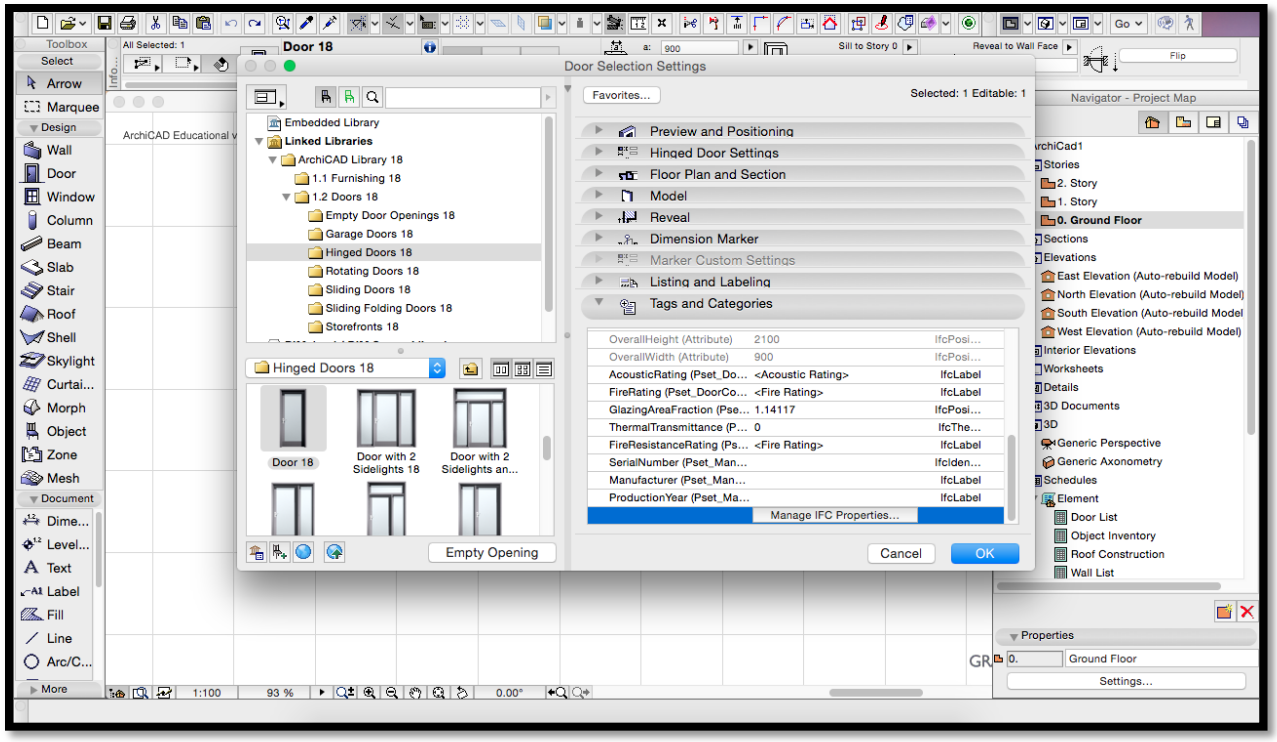
Information

Rule 12 checks if a door has the properties of SelfClosing, FireExit and SmokeStop regardless of the value (True/False). The checker reported 1 error on this rule.

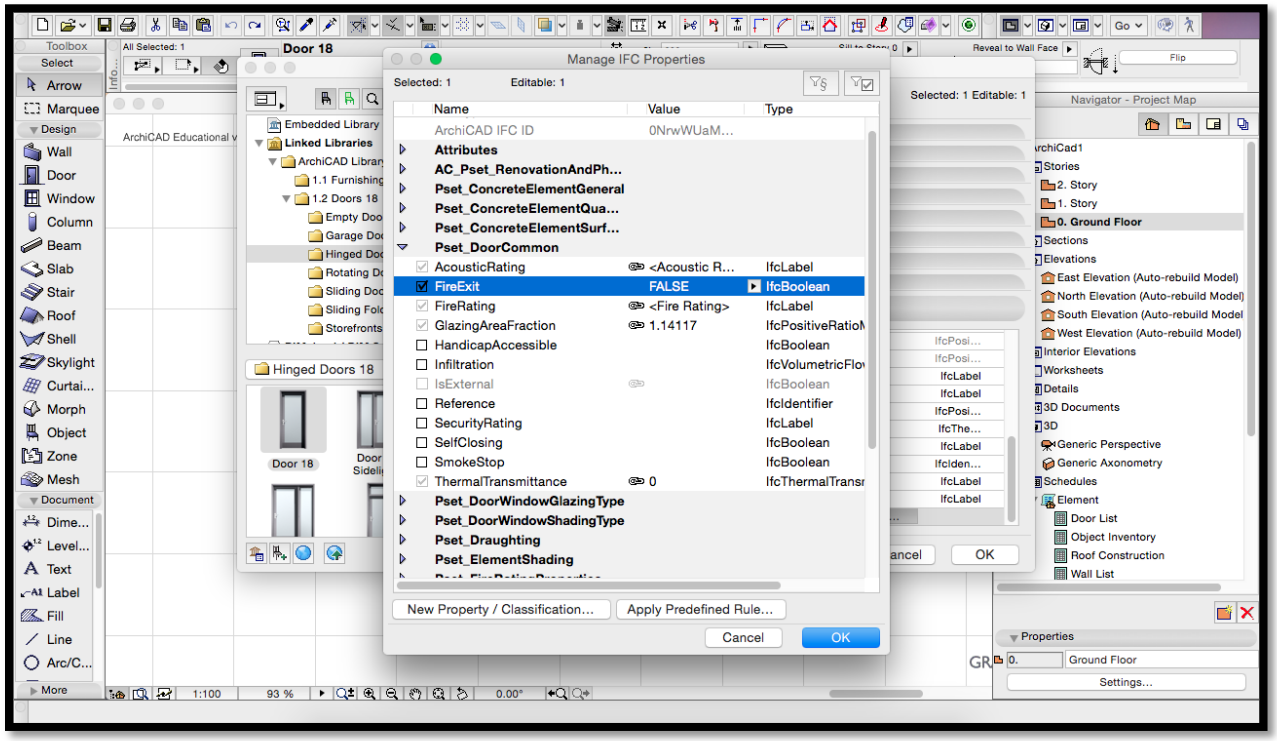
Modifications



Select the door and right-click it. Open the *Door Selection Settings*

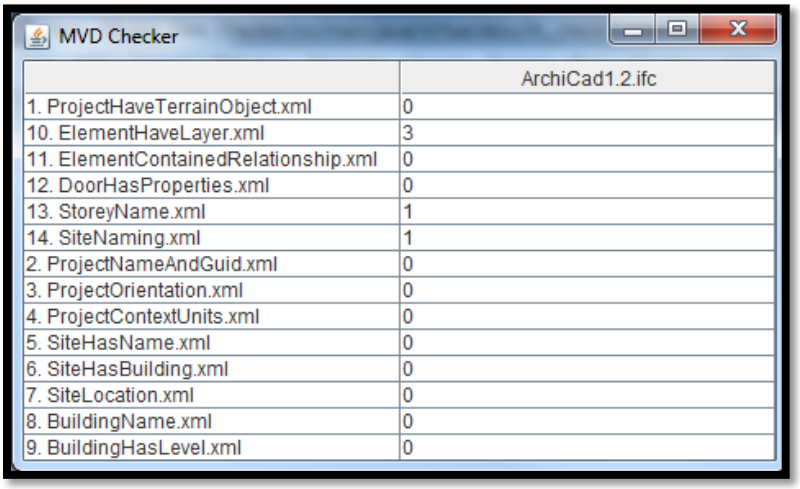


In the *Tags and Categories* tab, go to *Manage IFC Properties*..



Check the boxes of the attributes *FireExit, SelfClosing* and *SmokeStop* and enter their value's (TRUE/FALSE)

Results



	ArchiCad1.2.ifc
1. ProjectHaveTerrainObject.xml	0
10. ElementHaveLayer.xml	3
11. ElementContainedRelationship.xml	0
12. DoorHasProperties.xml	0
13. StoreyName.xml	1
14. SiteNaming.xml	1
2. ProjectNameAndGuid.xml	0
3. ProjectOrientation.xml	0
4. ProjectContextUnits.xml	0
5. SiteHasName.xml	0
6. SiteHasBuilding.xml	0
7. SiteLocation.xml	0
8. BuildingName.xml	0
9. BuildingHasLevel.xml	0

4.2.2.13. StoryName

Information

This rule checks if the name of the Story's follow the naming conventions given in the RGB BIM norm 2013 2.1.9.

RGB BIM norm 2013 2.1.9:

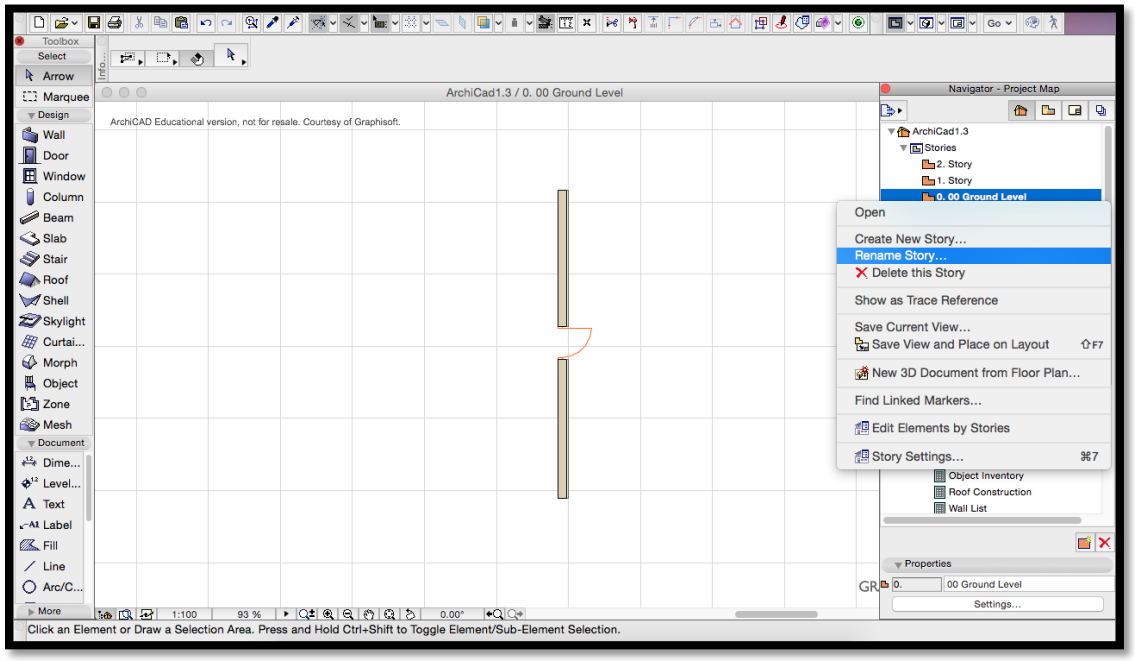
Example 1: *-1 Basement, 00 Ground level, 01 Second Floor*

When there is a split level a letter can be added

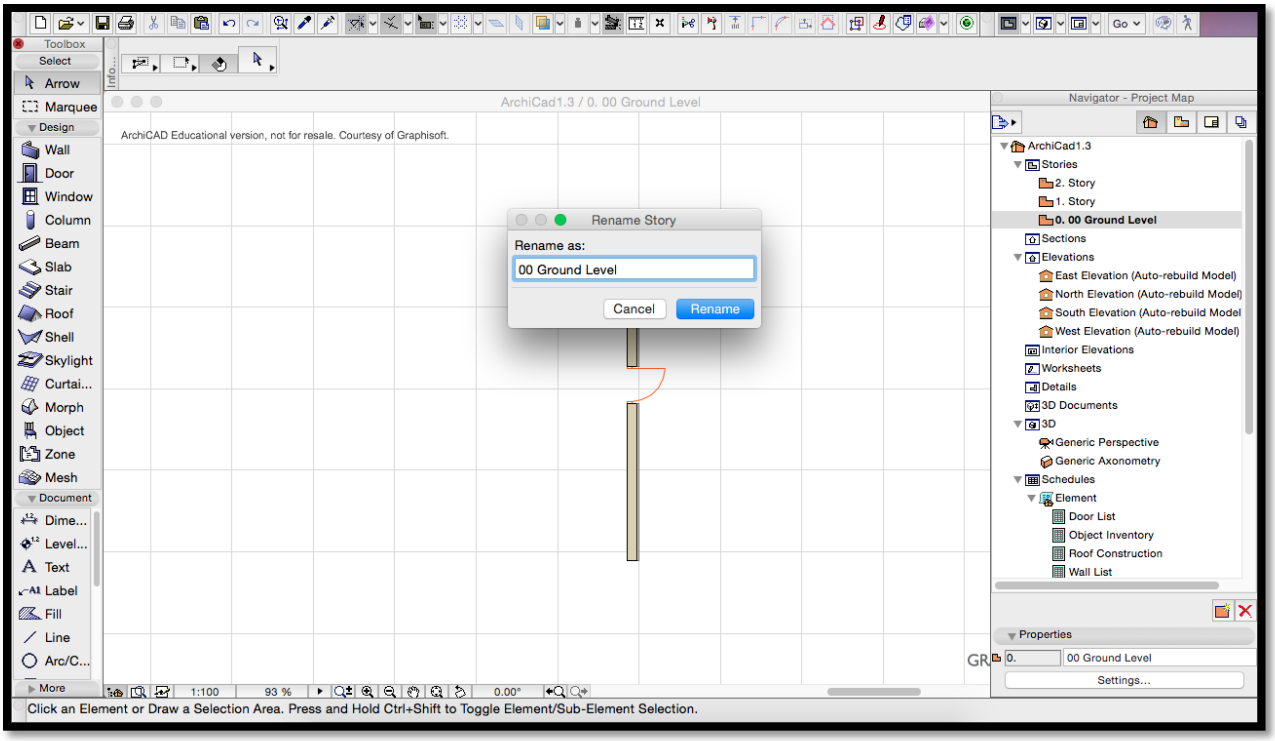
Example 2: *-1 Basement, 00 Ground level, 00a Split Level, 01 Second Floor*

On this rule one error was reported.

Modifications



Right-click on the Story for which you want to change the name and click *Rename Story…*



Change the name of the story into one that follows the rule and click *Rename*.

4.2.2.14. SiteNaming

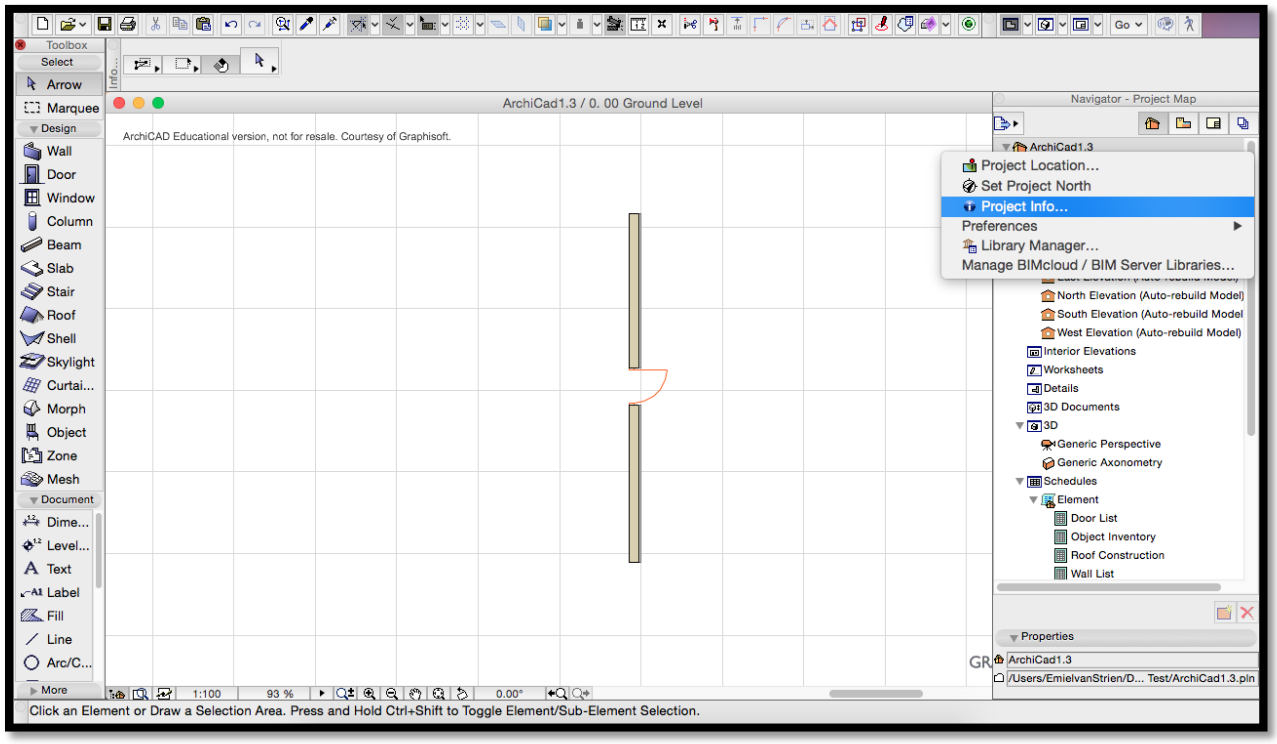
Information

This rule checks if the name of the Site follows the naming conventions given in the RGB BIM norm 2013 2.2.7.2.

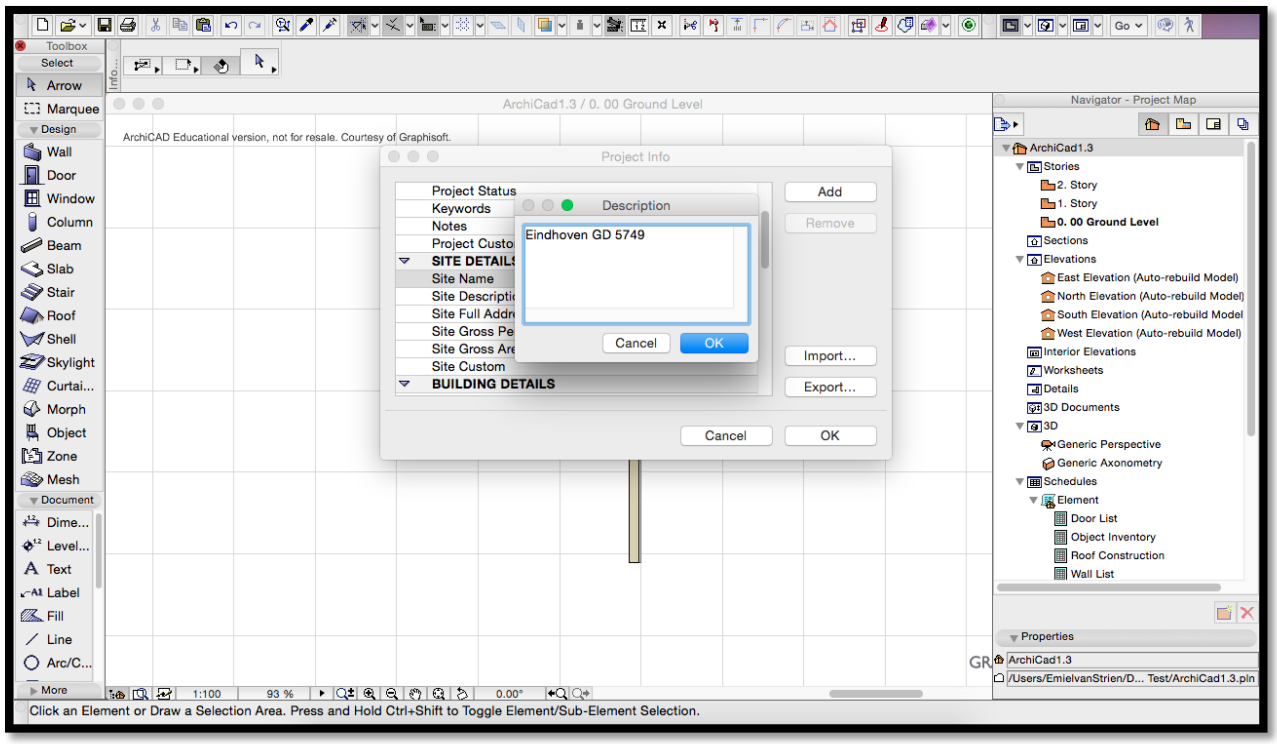
RGB BIM norm 2013 2.2.7.2:
Example 1: *Delft AB 1234*
When project is spread over multiple terrains:
Example 2: *Delft AB 1234 – Delft AC 1234*

On this rule one error was reported.

Modifications

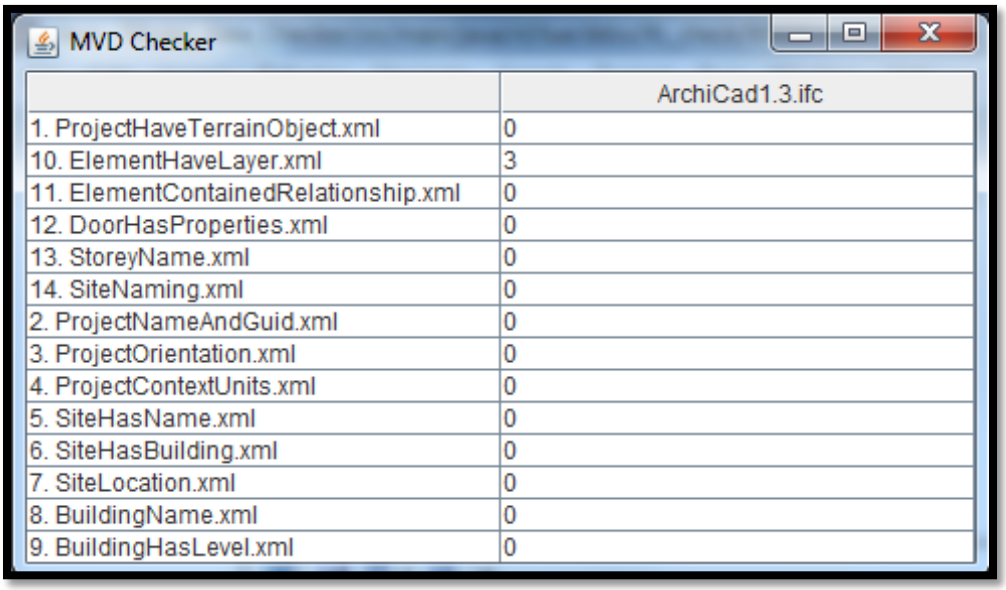


In the navigator on the right side right-click the *project button*. Then click on *Project Info…*



Change the *Site Name* into something that follows the rule and click *OK*

4.2.3 Result



	ArchiCad1.3.ifc
1. ProjectHaveTerrainObject.xml	0
10. ElementHaveLayer.xml	3
11. ElementContainedRelationship.xml	0
12. DoorHasProperties.xml	0
13. StoreyName.xml	0
14. SiteNaming.xml	0
2. ProjectNameAndGuid.xml	0
3. ProjectOrientation.xml	0
4. ProjectContextUnits.xml	0
5. SiteHasName.xml	0
6. SiteHasBuilding.xml	0
7. SiteLocation.xml	0
8. BuildingName.xml	0
9. BuildingHasLevel.xml	0

After the Revit Test the MVD Checker was fixed so it did support the Aggregation rule. Because of the rule 1, 6 and 9 do not give errors in this test.

Rule 10 still doesn't work correctly

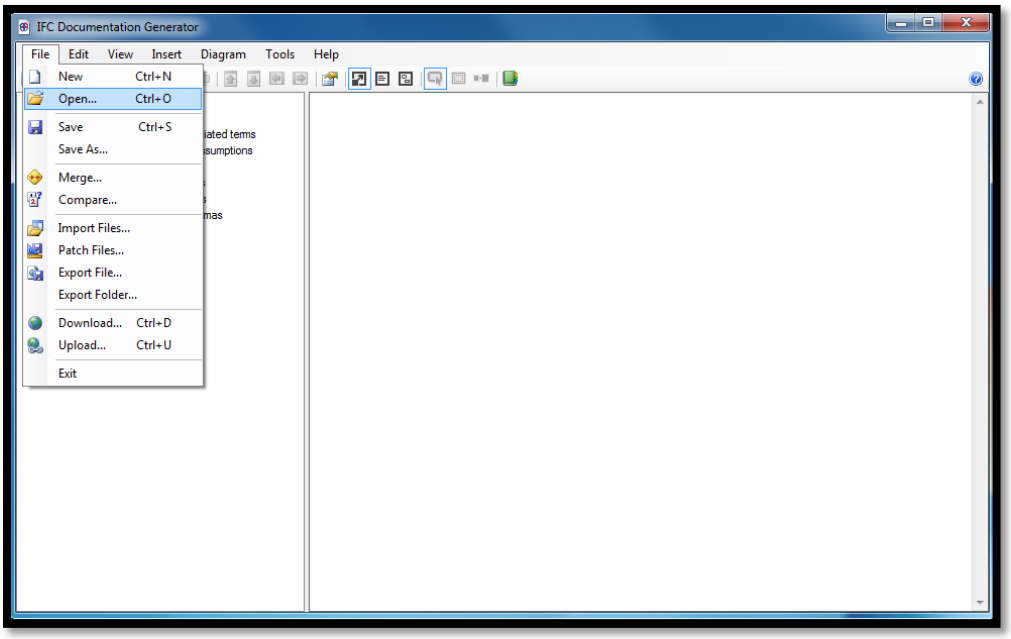
5. Creating new rules

5.1 IFCDoc

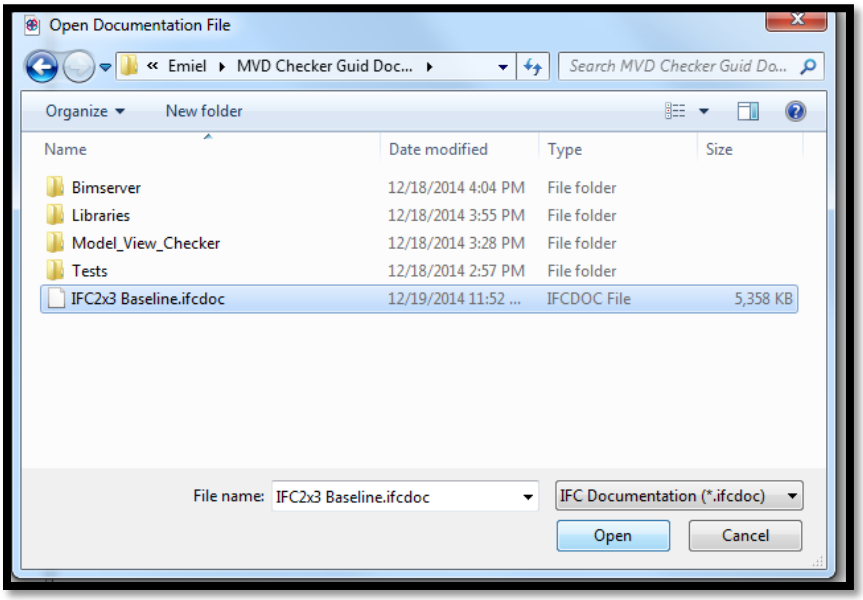
IFCDoc stands for IFC Documentation Generator. This is a tool to create MVDXML rules. You first need to download or copy a baseline file from the *MVD Checker Guide Documents folder.* The baseline file has different data schemas so you don't have to start from scratch. Since the MVD Checker supports IFC2x3 choose a baseline for IFC2x3.

You can download the baseline file at: http://www.buildingsmart-tech.org/specifications/specification-tools/IFCDoc-tool/IFCDoc-baselines

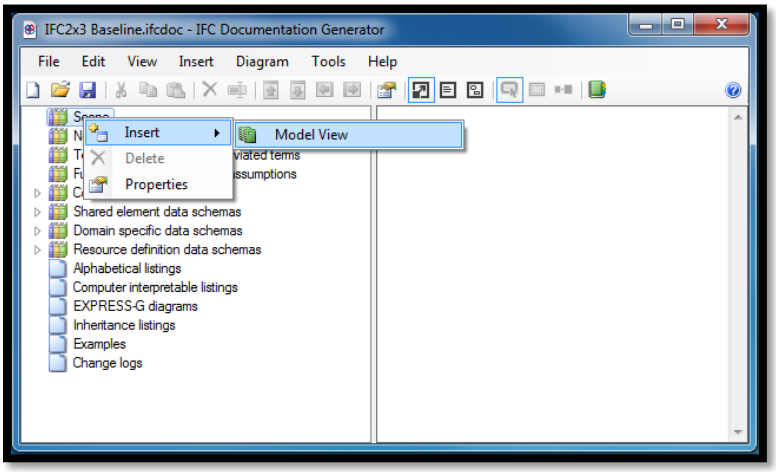
Go to *File →Open*



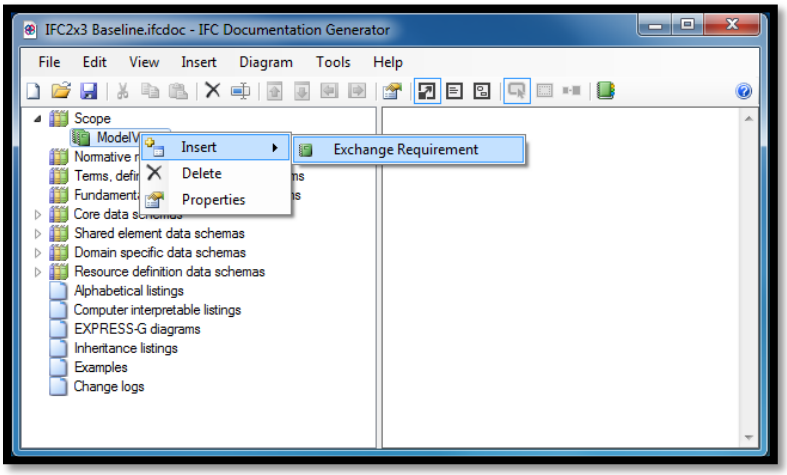
Open the Baseline file



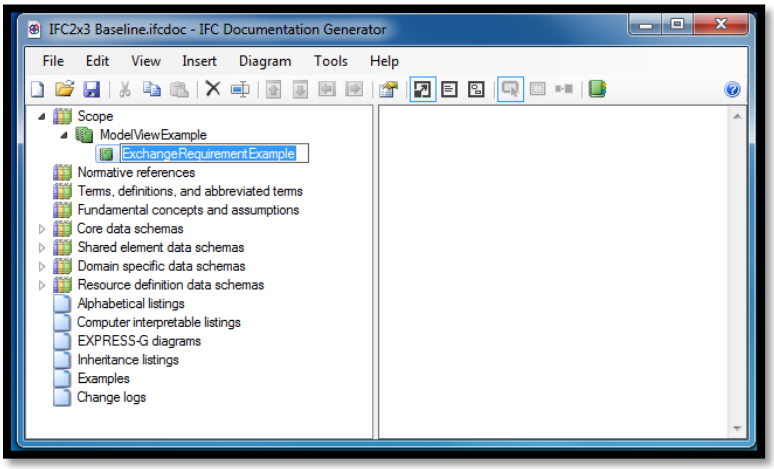
Insert a *Model View* in the *Scope*



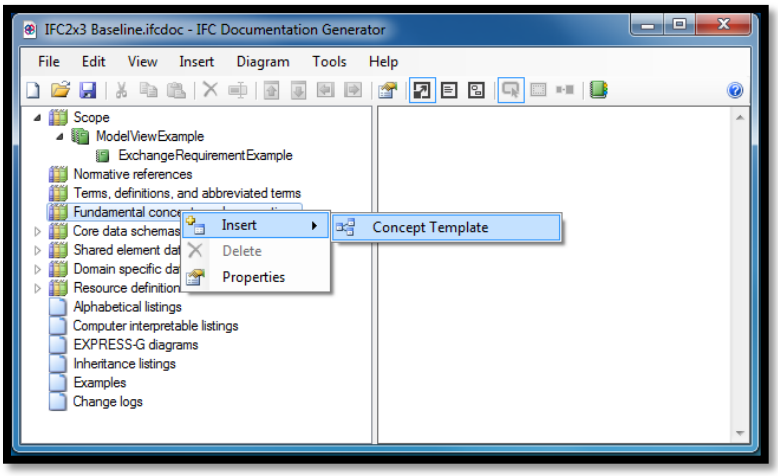
Then insert an *Exchange Requirement* in the *Model View*



You can rename both of them

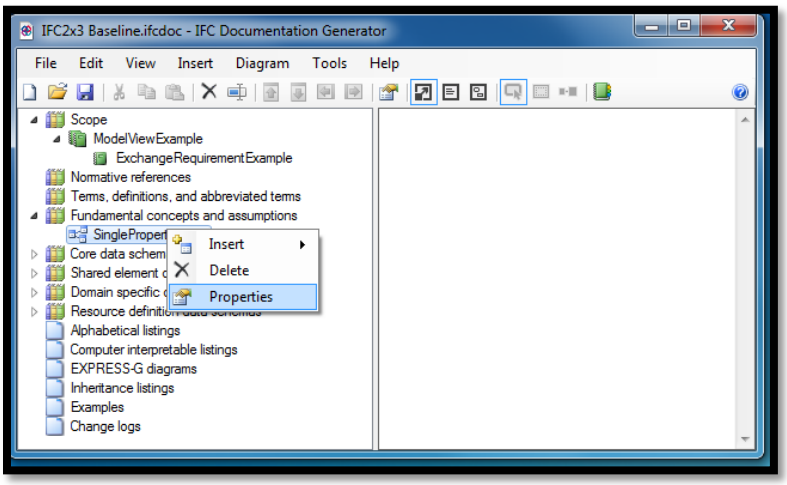


Then you create a Concept Template. You can later apply this Concept Template at Concept level. Insert a *Concept* Template in the *Fundamental concepts and assumptions*

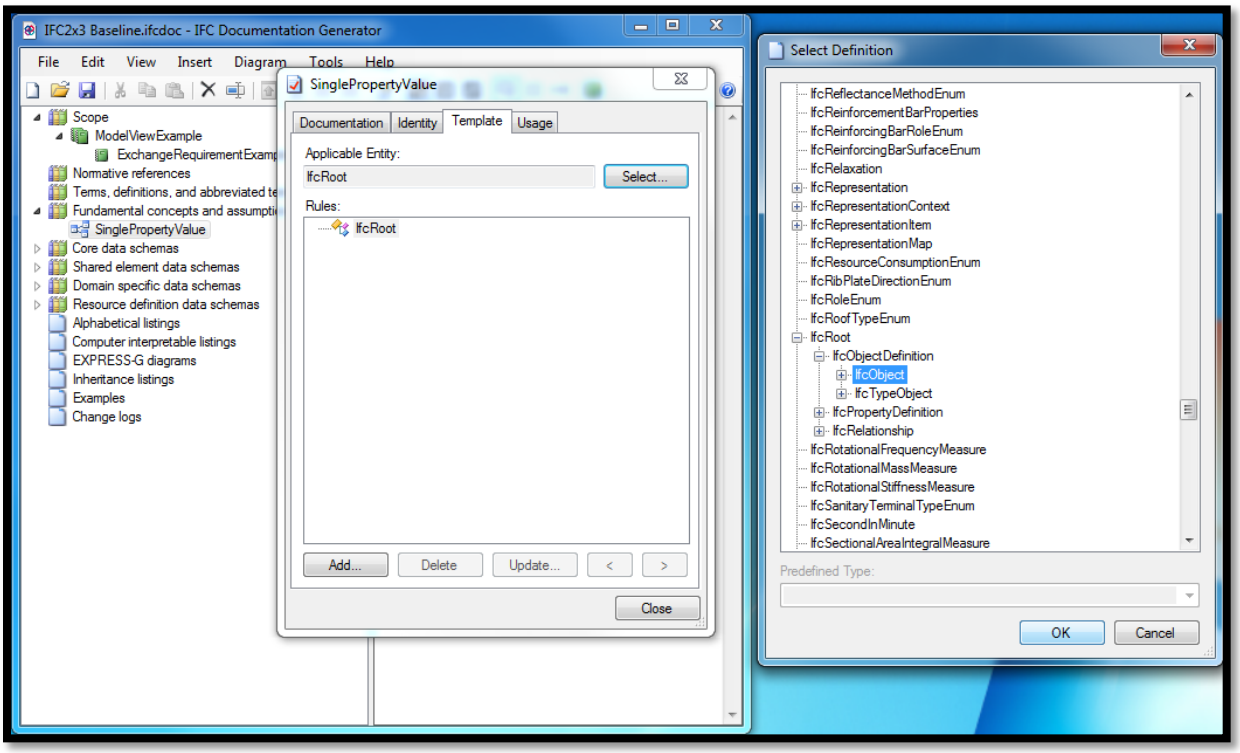


In this example a SinglePropertyValue Concept Template is created. This will let you create a rule in which you can assign SinglePropertyValues to all the subtypes of ifcObject.

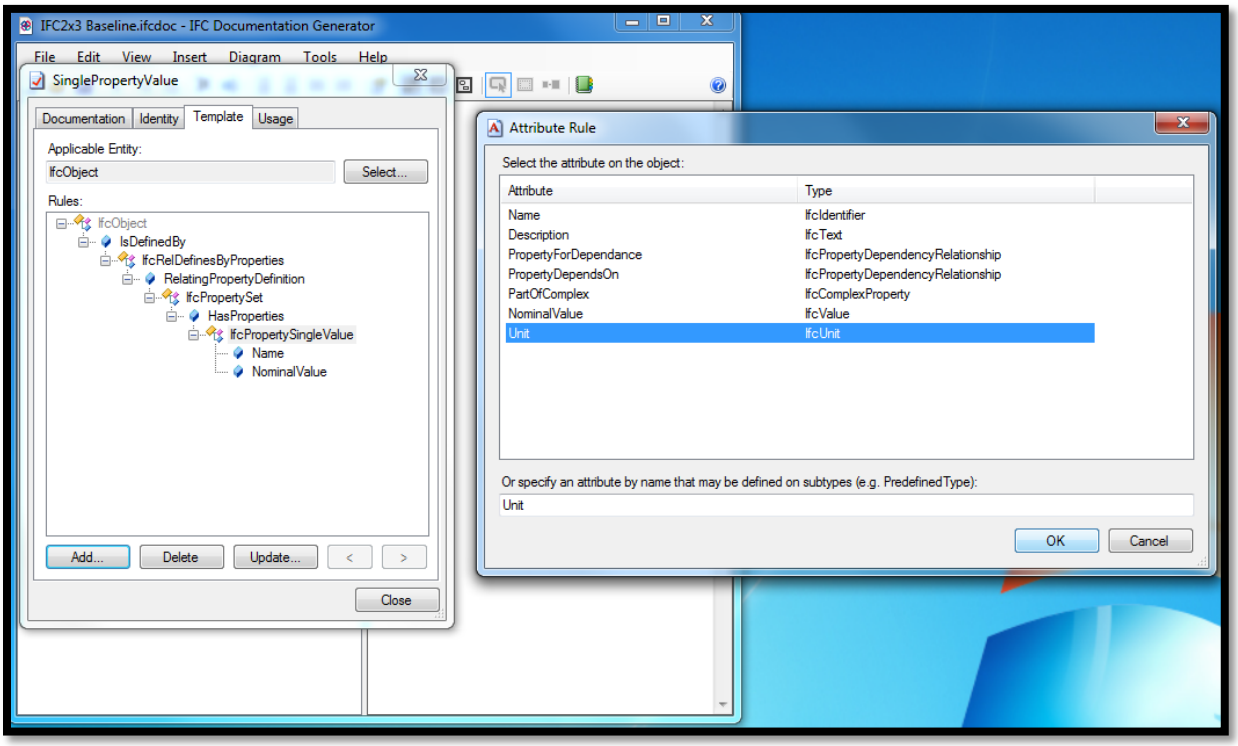
First rename the Concept Template and then go to *Properties*



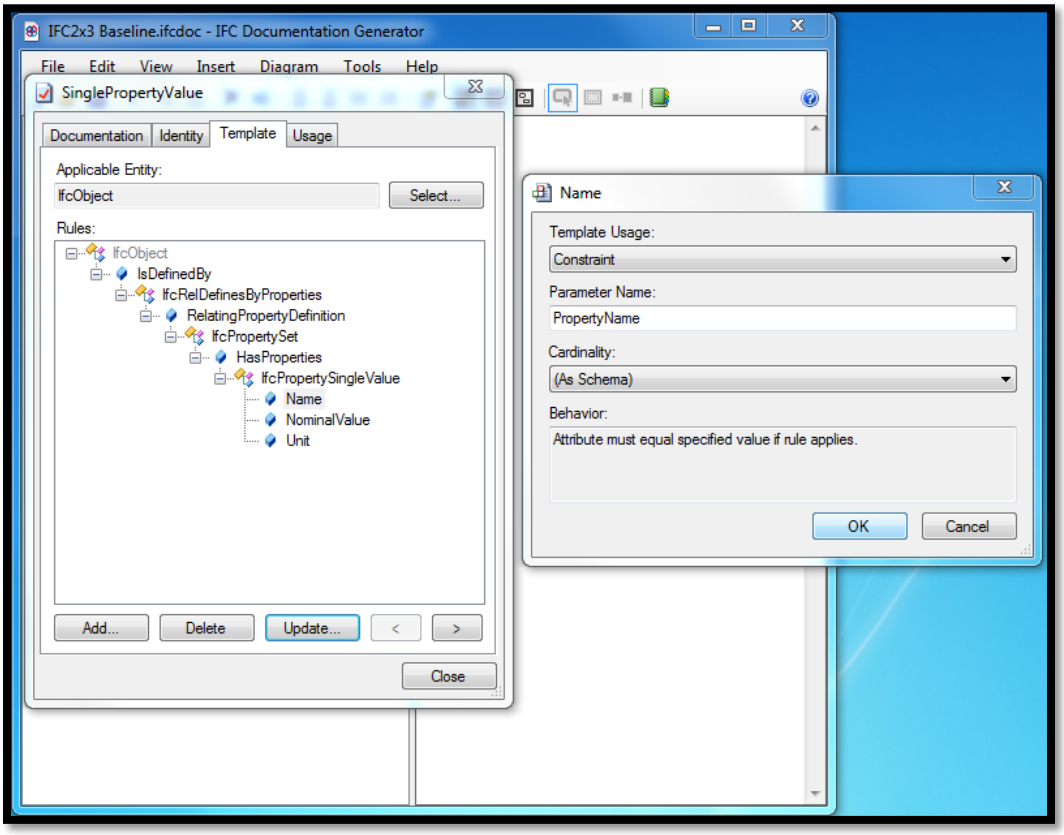
In the *Template* Tab you can create the structure of the Concept Template. First you *select* the Applicable Entity



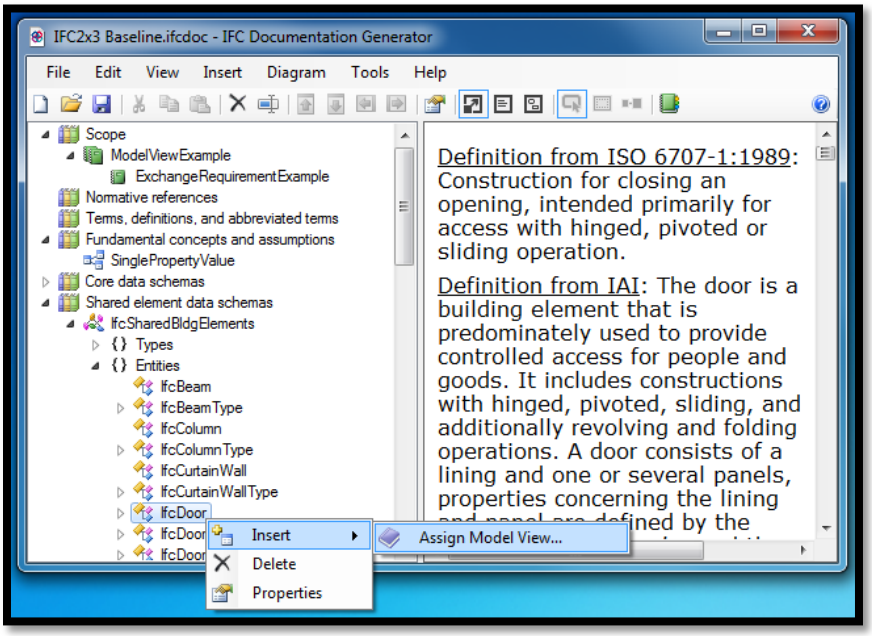
Add Entity and Attribute rules to ifcObject (the Applicable Entity) until the structure is complete



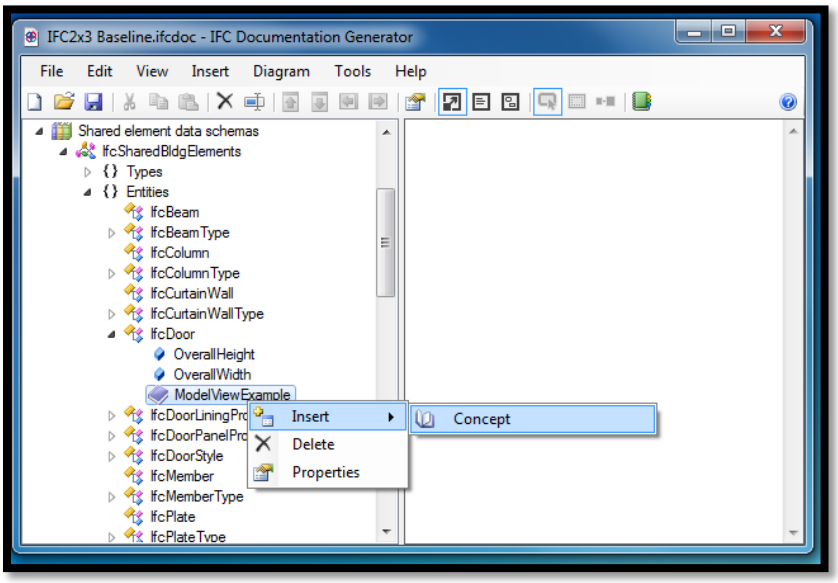
Then you can *Update* the attribute and entity rules to give them a cardinality or add parameters to them. You create a parameter to the rule by setting a *Constraint* and filling in the *Parameter Name.* On concept level this will let you change the name of the SinglePropertyValue (Here it's called PropertyName)



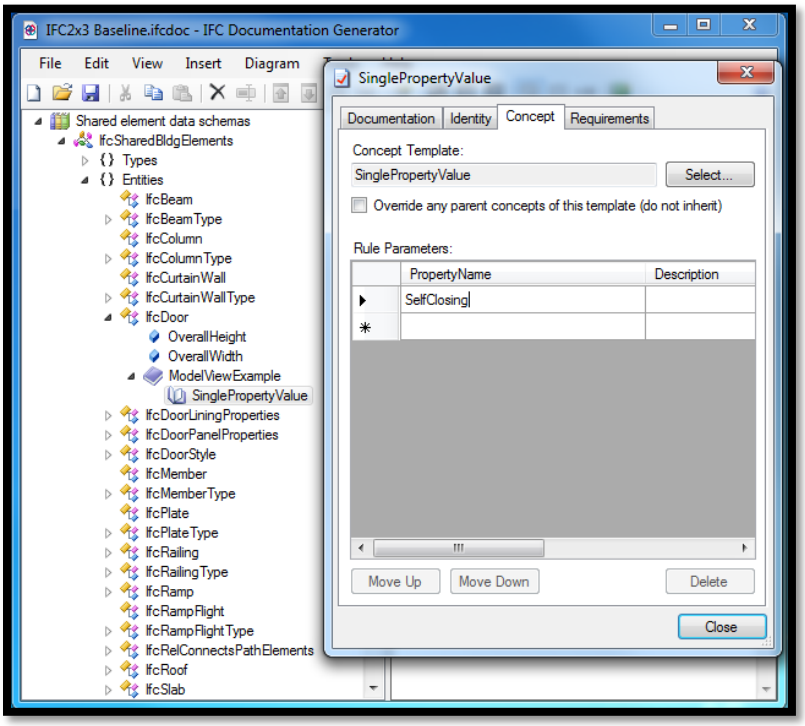
When the Concept Template is done you can add the Concept Template to a subtype of IfcObject. In this case it is added to IfcDoor, so a rule is created which checks if all IfcDoors have a certain property. First assign the ModelViewExample to the IfcDoor



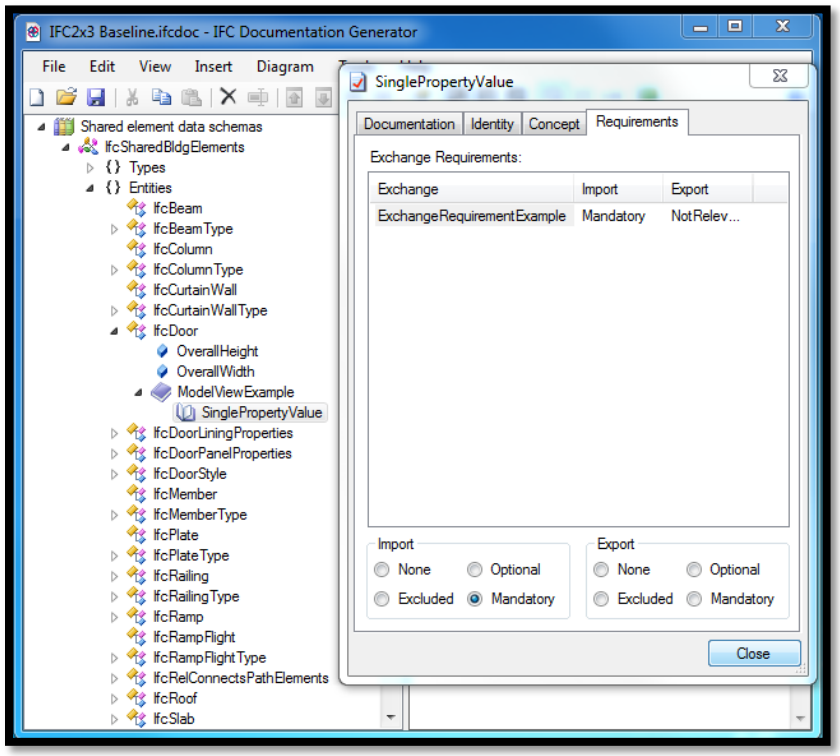
Assign a concept, in this case SinglePropertyValue, to the Model View



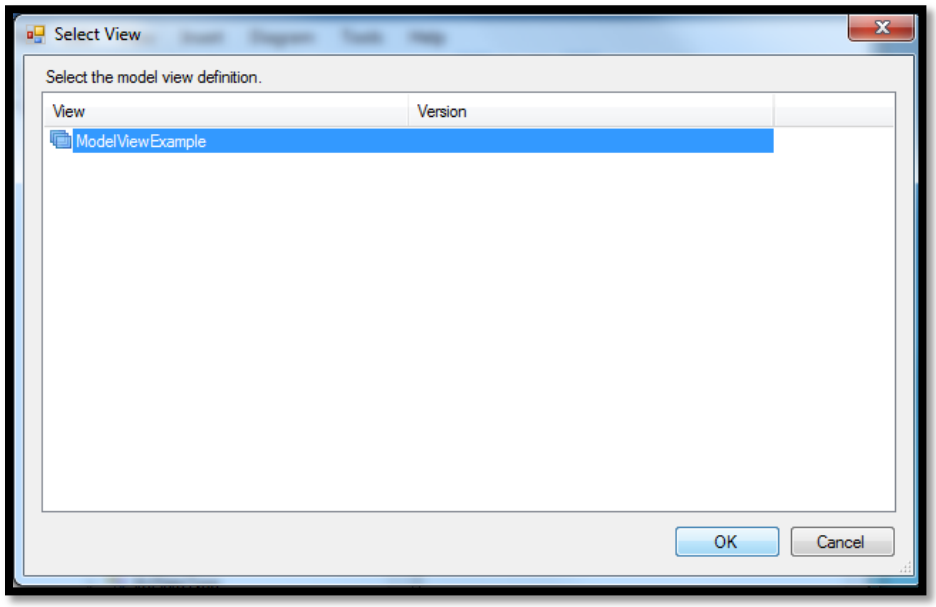
Then right click on the concept (SinglePropertyValue) and go to the *properties*. In the *Concept Tab* you can add a name to the PropertyName. In this case ifcDoor is going to be checked if it has the property of SelfClosing. So the propertyname is called “SelfClosing”



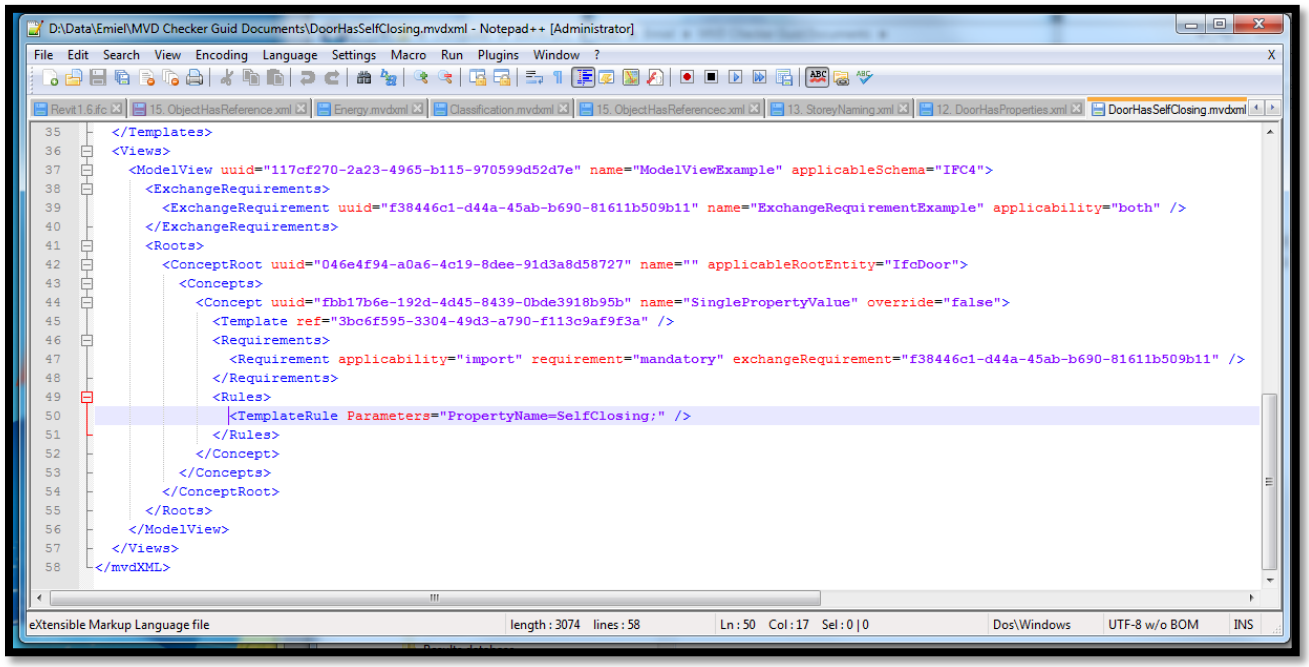
In the *Requirements Tab* Select the ExchangeRequirement and set it to *Mandatory* and then Close



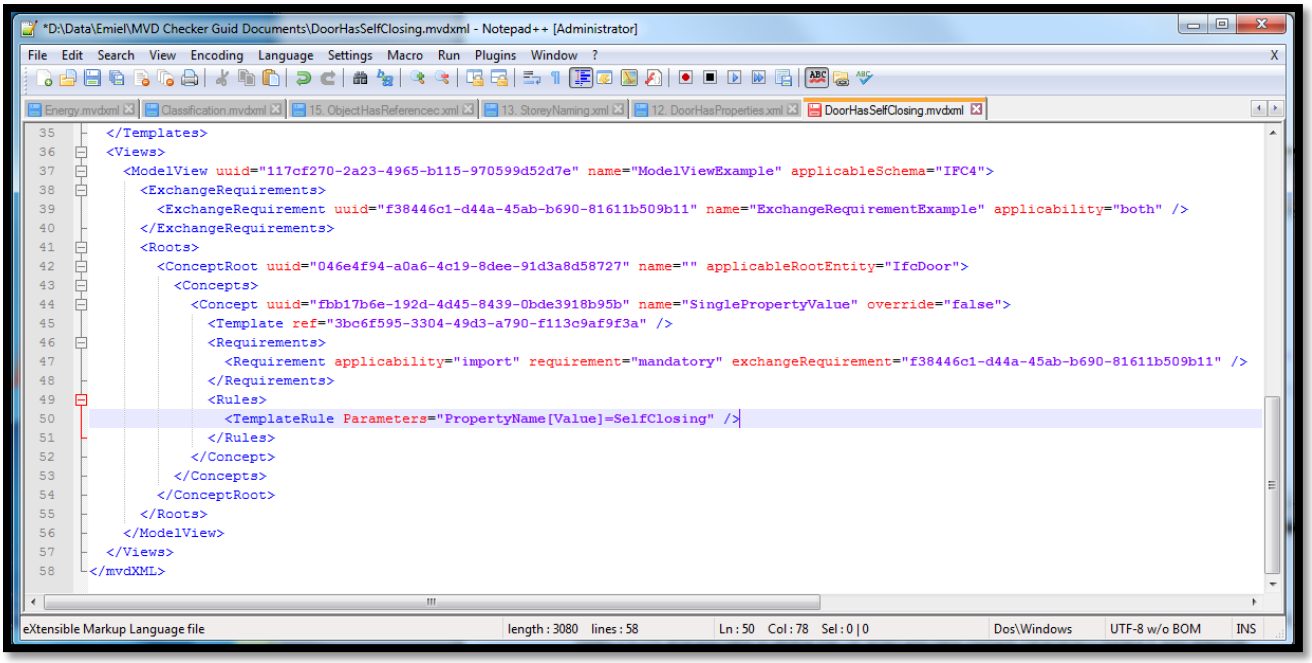
When this is done the file can be exported. *File→Export File.* Save it as a .mvdxml document. Select the Model View which needs to be exported. In this case ModelViewExample



When the exported .mvdxml file is opened you get an XML document with a Concept Template and the Model View you exported with the Concept. The Model View-part of the XML-rule looks like this:



Here you can see that the applicableRootEntity is an IfcDoor and that the TemplateRule has a Parameters=”PropertyName=SelfClosing;”
Before the MVD Checker can run this rule you have to add an extension to the Template rule. This can be: [Value], [Size], [Unique], [Type]. And you have to delete the ; or when you have multiple Parameters replace it with AND/OR. For more information about this you can check the XML1.1 documentation :
http://www.w3.org/TR/xml11/

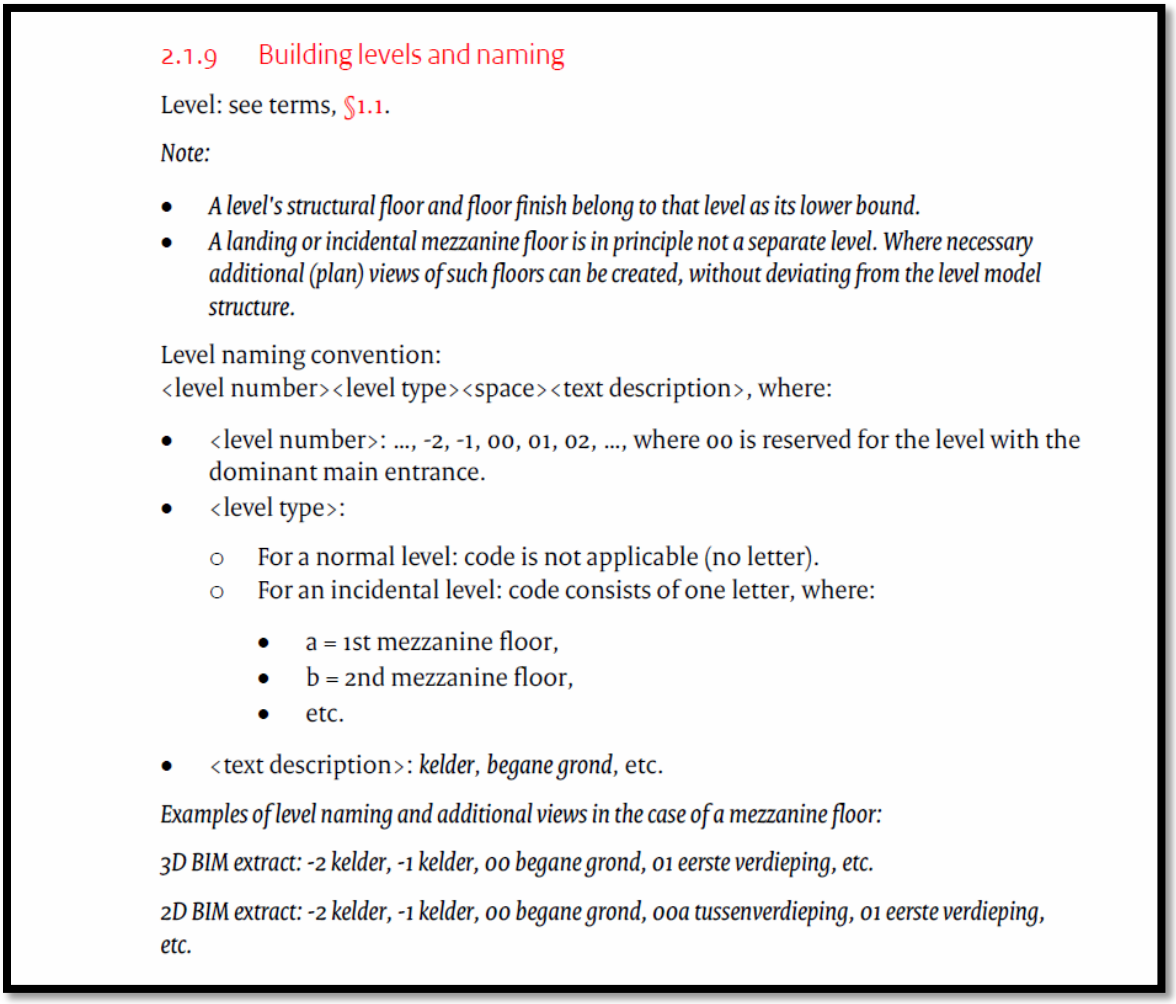


After this the XML rule is ready to be used in the MVD Checker.

5.2 Created and tested new rules

5.2.1 Rule 13 (StoryName)

This rule is based on the RGD BIM norm 2013 2.1.9 Building levels and naming.



2.1.9 Building levels and naming

Level: see terms, §1.1.

Note:

- *A level's structural floor and floor finish belong to that level as its lower bound.*
- *A landing or incidental mezzanine floor is in principle not a separate level. Where necessary additional (plan) views of such floors can be created, without deviating from the level model structure.*

Level naming convention:
<level number><level type><space><text description>, where:

- <level number>: ..., -2, -1, 00, 01, 02, ..., where 00 is reserved for the level with the dominant main entrance.
- <level type>:
  - For a normal level: code is not applicable (no letter).
  - For an incidental level: code consists of one letter, where:
    - a = 1st mezzanine floor,
    - b = 2nd mezzanine floor,
    - etc.
- <text description>: *kelder, begane grond*, etc.

Examples of level naming and additional views in the case of a mezzanine floor:

3D BIM extract: -2 kelder, -1 kelder, 00 begane grond, 01 eerste verdieping, etc.

2D BIM extract: -2 kelder, -1 kelder, 00 begane grond, 00a tussenverdieping, 01 eerste verdieping, etc.

This is the created rule in XML, with the Concept Template and the Model View with the Concept, using IFCDoc

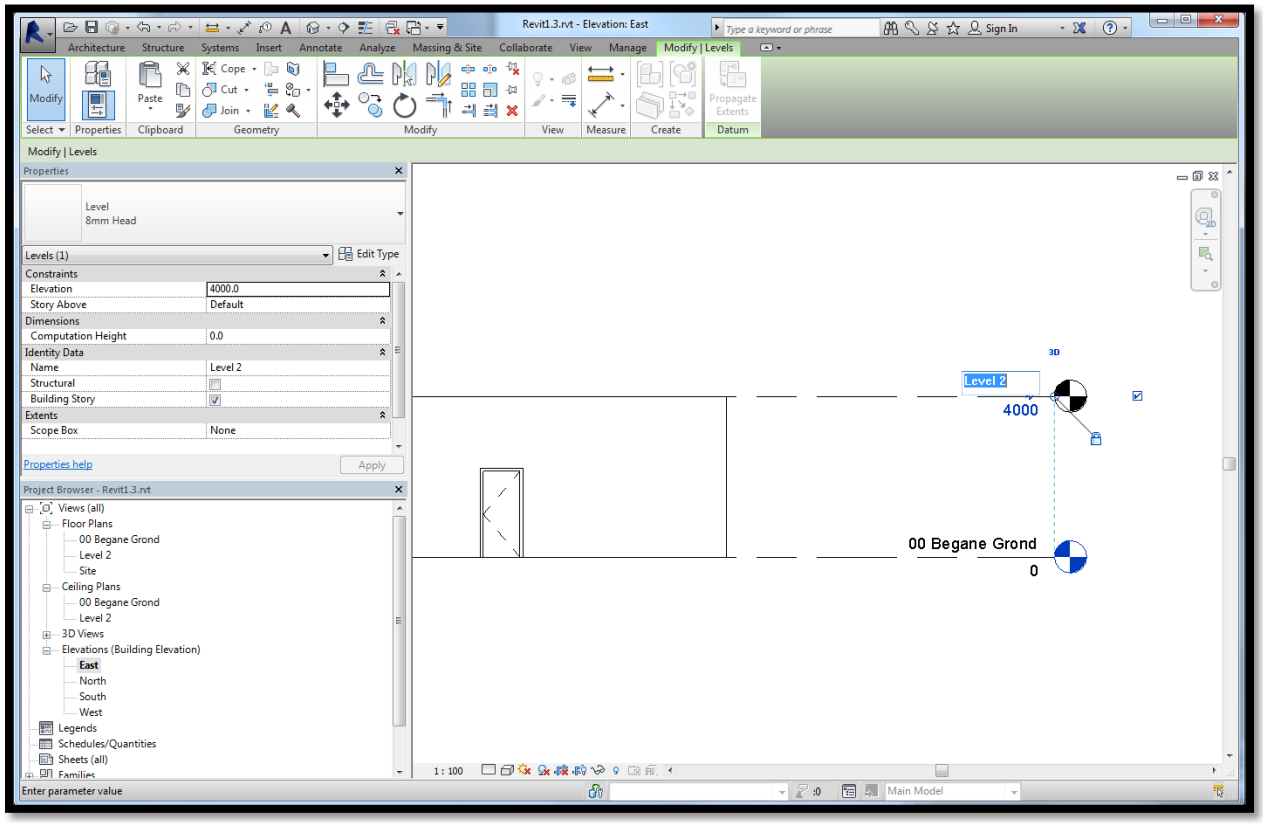
```xml
<?xml version="1.0"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="00000000-0000-0000-0000-000000000000"
    name="" xmlns="http://buildingsmart-tech.org/mvdXML/mvdXML1-1">
    <Templates>
        <ConceptTemplate uuid="5966aedc-339a-4d84-abc0-62f67dc3bbea"
            name="Identity" applicableSchema="IFC4" applicableEntity="IfcRoot">
            <Rules>
                <AttributeRule AttributeName="GlobalId" Cardinality="One" />
                <AttributeRule RuleID="Name" AttributeName="Name"
                    Cardinality="One" />
            </Rules>
        </ConceptTemplate>
    </Templates>
    <Views>
        <ModelView uuid="dce6f5c2-cd73-4c6d-9a13-690be9adf597" name="Rqd"
            applicableSchema="IFC4">
            <ExchangeRequirements>
                <ExchangeRequirement uuid="7d3dd1bf-37b0-4c50-bf6e-7ec4d7a6e52f"
                    name="Rqd standard" applicability="both" />
            </ExchangeRequirements>
            <Roots>
                <ConceptRoot uuid="dfcf1787-3b31-4c46-98c9-155f76a5f82a"
                    name="" applicableRootEntity="IfcBuildingStorey">
                    <Concepts>
                        <Concept uuid="599e253d-91ca-48b1-af34-df3ae6136f28" name="Identity"
                            override="false">
                            <Definitions>
                                <Definition>
                                    <Body><![CDATA[This concept checks the name of the storey should follow the level
                                    naming convention <level number><level type><space><text description>
                                    RGD Norm 2.1.9 Building levels and naming]]>
                                    </Body>
                                </Definition>
                            </Definitions>
                            <Template ref="5966aedc-339a-4d84-abc0-62f67dc3bbea" />
                            <Requirements />
                            <Rules>
                                <TemplateRule
                                    Parameters="Name[Value]=reg '(-|[0-9])[0-9] [a-zA-Z ]*[a-z]?'" />
                            </Rules>
                        </Concept>
                    </Concepts>
                </ConceptRoot>
            </Roots>
        </ModelView>
    </Views>
</mvdXML>
```

When the Revit1.3.IFC-File is checked it reports an error on this rule.

When manually checked in the IFC-File the name of the BuildingStorey isn't according to the rule but is: 'Level 1'

```
#98= IFCBUILDINGSTOREY('24HIdRGyjAxxBqaKbjoqjo',#41,'Level 1',$,$,#96,$,'Level 1',.ELEMENT.,0.);
```

In Revit you can change the name of the BuildingStorey by going to one of the Elevations and simply change the name of all the BuildingStoreys.



In the IfcFile it looks like this:

```
82  #98= IFCBUILDINGSTOREY('24HIdRGyjAxxBqaKbjoqjo',#41,'00 Begane Grond',$,$,#96,$,'00 Begane Grond',.ELEMENT.,0.);
```

After checking Revit1.5.IFC-File it didn't report an error on this rule.

5.2.2 Rule 14 (SiteNaming)

This rule is based on the RGD BIM norm 2013 2.2.7.2 Terrein.

2.2.7.2 Terrein

- **Beschrijving**: het terrein, de topografische site van het project.
- **Geometrie**: 3D
  Het IFC-object heeft de driedimensionale vorm en afmetingen van tenminste het terreinoppervlak van het project:
  - Contour: tenminste van het kadastrale perceel of het geheel van percelen.
  - Hoogte: volgens de plaatselijke topografie.
- **Locatie**: het referentiepunt van het terrein in het IFC-model is geografisch correct gepositioneerd.
- **Relaties**: het project omvat slechts één terreinobject.
- **IfcObject**: IfcSite

  Attributes:
  - **Name**: <kadastrale aanduiding>
    Voorbeeld: Delft AB 1234
    Indien het project zich over meerdere kadastrale percelen uitstrekt, zijn alle kadastrale aanduidingen opgegeven, onderling gescheiden door <spatie>"-"<spatie>.
    Voorbeeld: Delft AB 1234 - Delft AB 1235
  - **RefLatitude**: geografische locatie, breedtegraad op referentiepunt volgens WGS84.
  - **RefLongitude**: geografische locatie, lengtegraad op referentiepunt volgens WGS84.
  - **RefElevation**: geografische locatie, peil op referentiepunt, hoogtepeil t.o.v. NAP.

This is the created rule in XML, with the Concept Template and the Model View with the Concept, using IFCDoc

```xml
<?xml version="1.0"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="00000000-0000-0000-0000-000000000000">
  <Templates>
    <ConceptTemplate uuid="4f03fc6a-2a16-4fa5-b09e-cd0e711b5f7f" name="Naming" applicableSchema="IFC4" applicableEntity="IfcRoot">
      <Rules>
        <AttributeRule RuleID="Name" AttributeName="Name" Cardinality="One" />
        <AttributeRule AttributeName="GlobalId" Cardinality="One" />
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="af8f6312-2561-4181-808f-592a4c5e17ee" name="Example" applicableSchema="IFC4">
      <ExchangeRequirements>
        <ExchangeRequirement uuid="15e81157-a529-4931-abc6-18589f8665b9" name="Test" applicability="both" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="ee66a3b5-7016-49b8-b73f-3563a72c2723" name="" applicableRootEntity="IfcSite">
          <Concepts>
            <Concept uuid="c11e4b17-3c19-4ef6-b0fe-1e7698a69f20" name="Naming" override="false">
              <Template ref="4f03fc6a-2a16-4fa5-b09e-cd0e711b5f7f" />
              <Requirements>
                <Requirement applicability="import" requirement="mandatory" exchangeRequirement="15e81157-a529-4931-abc6-18589f8665b9" />
              </Requirements>
              <Rules>
                <TemplateRule Parameters="Name[Value]=reg '([A-Za-z ]+ [A-Z]{2} [0-9]{4} - )*[A-Za-z ]+ [A-Z]{2} [0-9]{4}'" />
              </Rules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

Since the project is able stretch out over multiple grounds and the naming should follow the rules of the RGB norm 2.2.7.2. 3 different IFC-Files are created by manually changing the IFC-File:

One which is stretched out over 2 grounds(Revit1.6.ifc)

```
309  #498= IFCSITE('24HIdRGyjAxxBqaKcID8UF',#41,'Delft AB 1234 - Den Bosch CB 1234'
```

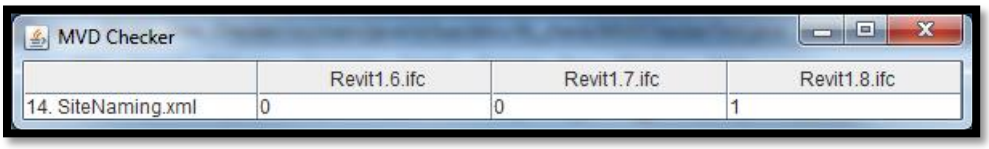
And one which is just on 1 ground (Revit1.7.ifc)

```
309  #498= IFCSITE('24HIdRGyjAxxBqaKcID8UF',#41,'Delft AB 1234'
```

And one which is on 1 ground, but doesn't follow the naming rules (Revit1.8.ifc)

```
309  #498= IFCSITE('24HIdRGyjAxxBqaKcID8UF',#41,'TU/E campus'
```

When tested, as expected, Revit1.6 and Revit1.7 didn't report an error, but Revit1.8 did.

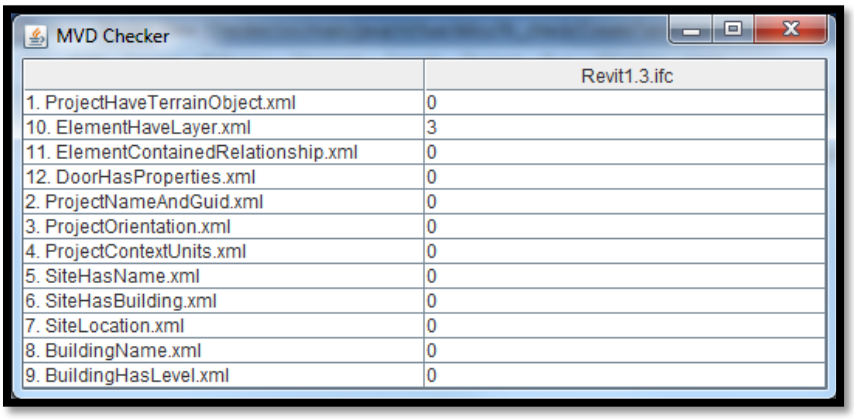


	Revit1.6.ifc	Revit1.7.ifc	Revit1.8.ifc
14. SiteNaming.xml	0	0	1

6. MVD Checker improvements

6.1 Fixing the aggregation rule

After finding out that all of the rules that used the Aggregation concept template gave errors multiple test were done involving reproducing and changing the rule using IFCDoc. The new rules had the same problem and kept producing errors. After some test and changes to the MVD checker it didn't create an error anymore. This meant that rule 1, 6 and 9 are also fixed.



	Revit1.3.ifc
1. ProjectHaveTerrainObject.xml	0
10. ElementHaveLayer.xml	3
11. ElementContainedRelationship.xml	0
12. DoorHasProperties.xml	0
2. ProjectNameAndGuid.xml	0
3. ProjectOrientation.xml	0
4. ProjectContextUnits.xml	0
5. SiteHasName.xml	0
6. SiteHasBuilding.xml	0
7. SiteLocation.xml	0
8. BuildingName.xml	0
9. BuildingHasLevel.xml	0

To make sure it checks the correct things some manual changes were made to the IFC-File.

These are the 3 lines in the IFC-File which create the link between the different IfcObjects.

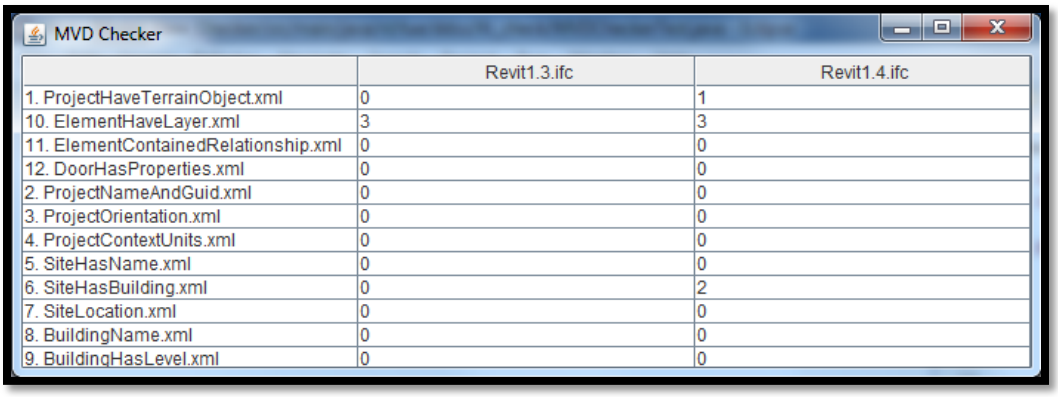
```
332  #545= IFCRELAGGREGATES('3Bw8hkJev9SgF8fsF$d0Ts',#41,$,$,#79,(#498));
333  #549= IFCRELAGGREGATES('1xWY5gkGL2BfNOJDsXe00e',#41,$,$,#498,(#89));
334  #553= IFCRELAGGREGATES('3ioAY2VOL1QOEtMGyzl4Rh',#41,$,$,#89,(#98));
```

#545 says that the IfcProject (#79) should have a IfcSite (#498)
#549 says that the IfcSite (#498) should have a IfcBuilding (#89)
#553 says that the IfcBuilding (#89) should have a IfcStorey (#98)

Multiple changed were made to a copy of the Revit1.3.IFC-File and saved as Revit1.4.ifc. These are the changes that were made.

- Line #549 was deleted so the IfcSite isn't linked to the IfcBuilding anymore.
- One extra IfcSite was added to the IFC-File and also to line #545 what should give one error since an IfcProject is only allowed to have one IfcSite

```
333  #545= IFCRELAGGREGATES('3Bw8hkJev9SgF8fsF$d0Ts',#41,$,$,#79,(#498,#499));
334  #553= IFCRELAGGREGATES('3ioAY2VOL1QOEtMGyzl4Rh',#41,$,$,#89,(#98));
```



	Revit1.3.ifc	Revit1.4.ifc
1. ProjectHaveTerrainObject.xml	0	1
10. ElementHaveLayer.xml	3	3
11. ElementContainedRelationship.xml	0	0
12. DoorHasProperties.xml	0	0
2. ProjectNameAndGuid.xml	0	0
3. ProjectOrientation.xml	0	0
4. ProjectContextUnits.xml	0	0
5. SiteHasName.xml	0	0
6. SiteHasBuilding.xml	0	2
7. SiteLocation.xml	0	0
8. BuildingName.xml	0	0
9. BuildingHasLevel.xml	0	0

This creates one error on rule 1 as expected since a Project is only allowed one IfcSite. On rule 6 it now creates 2 errors. One error is because the link between the site and the building was deleted and the other error appears because one extra site was added and this site also doesn't have a link with an IfcSite. So this rule now works correctly.