**Create a new VariaMos model**

May 2019

The VariaMos team

**Stage 0 - Pre-requirements**

*The pre-requirements are the things you need to actually create a new VariaMos model*

* Download, install and run VariaMos web **“development”** version. <https://github.com/SPLA/VARIAMOS-WEB/tree/development>

**Note:** in the following guides there will be a more detailed explanation on how to set up **VariaMos-Web**

* VariaMos Front-end ***(Remember: install the “development” version):*** <https://docs.google.com/document/d/1gU18SYM6A7M61e54O7L6jpvQpdQAhTZahMaPlWxrejk/edit>

**Stage 1 - Create a new model file**

Go tosrc/assets/js/models/custom/ and create a file called mymodel.js with the next content. This a new model that reuses the component model elements.

|  |
| --- |
| var mymodel\_main = function mymodel\_main(graph)  {  mymodel\_constraints(graph);  var data=[];  data["m\_type"]="normal" //custom type  data["m\_elements"]=mymodel\_elements(); //custom elements  data["m\_attributes"]=cmymodel\_attributes(); //custom attributes  data["m\_relations"]=null; //custom relations  data["m\_properties\_styles"]=null; //custom properties styles  data["m\_labels"]=null; //custom labels  data["m\_clon\_cells"]=null; //custom clon cells  data["m\_constraints\_ic"]=null; //custom constraints in element creation  data["m\_overlay"]=null; //custom overlays  return data;    function mymodel\_constraints(graph){  graph.multiplicities=[]; //reset multiplicities  graph.multiplicities.push(new mxMultiplicity(  true, "component", null, null, 0, 0, null,  "Invalid connection",  "Only shape targets allowed"));  graph.multiplicities.push(new mxMultiplicity(  true, "file", null, null, 0, 1, ["component"],  "Only 1 target allowed",  "Only shape targets allowed"));  }  function mymodel\_elements(){  var component = {src:projectPath+"images/models/component/component.png", wd:100, hg:40, style:"shape=component", type:"component", pname:"Component"};  var file = {src:projectPath+"images/models/component/file.png", wd:100, hg:40, style:"shape=file", type:"file", pname:"File"};  var elements=[];  elements[0]=component;  elements[1]=file;    return elements;  }  function mymodel\_attributes(){  var attributes=[];  attributes[0]={  "types":["file"],  "custom\_attributes":[{  "name":"filename",  "def\_value":""  },  {  "name":"destination",  "def\_value":""  }]  };    return attributes;  }    }  export default mymodel\_main |

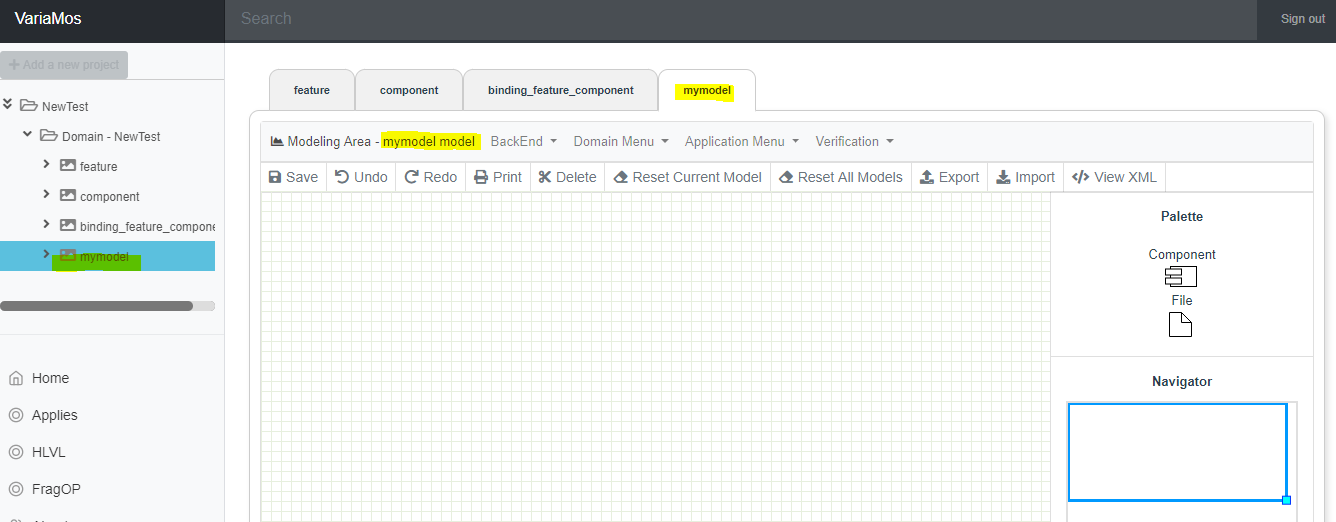
**Stage 2 - Modify global\_info.js**

Go tosrc/assets/js/common/global\_info.js and include the previous model. Check the highlighted elements.

|  |
| --- |
| function getModelInfo(){  var info =[];  //list of graphical models  info["gmodels"]=["feature","component","binding\_feature\_component","mymodel"];  //define feature model main info  info["feature"]={projFolders:["Domain","Application","Adaptation"]};  //define component model main info  info["component"]={projFolders:["Domain"]};  //define binding model main info  info["binding\_feature\_component"]={projFolders:["Domain"]};  //define mymodel info  info["mymodel"]={projFolders:["Domain"]};  return info;  } |

**Stage 3 - Create a new project.**

The last step, is to open VariaMos, and create a new project. Then navigate to the domain folder and you will see how it shows the new model option.

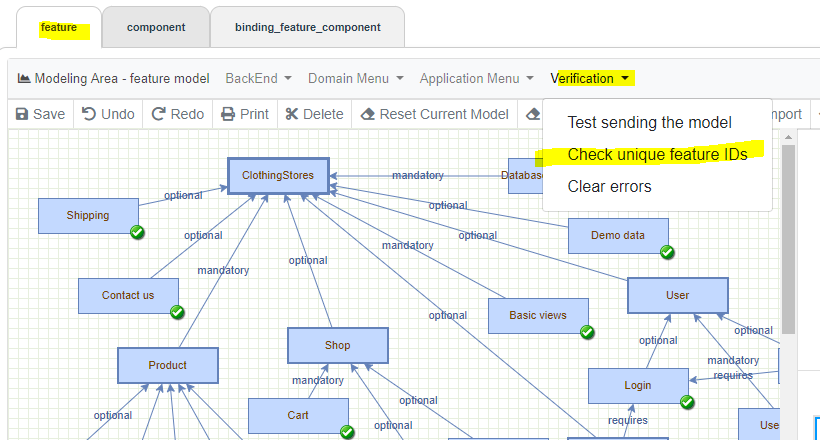


.

**Extra - Custom model verifications**

If you want to include custom model verifications. You just need to create a folder and a file called src/assets/js/models/custom/mymodel/verification/mymodel.js and reuse the content from the src/assets/js/models/custom/feature/verification/feature.js (it shows how to include custom verifications).

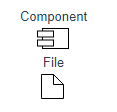
Those verification will appear automatically in this section:



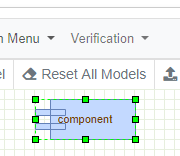
**Extra - Custom model shapes**

Each model element has two types of shapes that must be designed.

1. **The graphical shape:** which is a png image that can be designed in draw.io.



1. **The svg shape:** which is the shape that appears in the model area.



Both shapes are defined in the model file in this section:

|  |
| --- |
| function mymodel\_elements(){  var component = {src:projectPath+"images/models/component/component.png", wd:100, hg:40, style:"shape=component", type:"component", pname:"Component"};  var file = {src:projectPath+"images/models/component/file.png", wd:100, hg:40, style:"shape=file", type:"file", pname:"File"}; |

* The graphical shape must be stored in this path public/images/models/mymodel/shape\_name.png
* The svg shape must be defined in public/xml/MX/custom\_shapes.xml

**Note:** there are some predefined svg shapes that MxGraph already supports: <https://jgraph.github.io/mxgraph/docs/js-api/files/shape/mxShape-js.html>

If you want to develop the svg from scratch (in the custom\_shapes.xml file), you can duplicate the current shapes that are available in that file, as an example.

**CUSTOM SHAPE DATA:**

[https://about.draw.io/shape-styles/](https://hangouts.google.com/_/elUi/chat-redirect?dest=https%3A%2F%2Fabout.draw.io%2Fshape-styles%2F)

**Custom SVG elements:**

[https://jgraph.github.io/mxgraph/docs/js-api/files/shape/mxStencil-js.html#mxStencil](https://hangouts.google.com/_/elUi/chat-redirect?dest=https%3A%2F%2Fjgraph.github.io%2Fmxgraph%2Fdocs%2Fjs-api%2Ffiles%2Fshape%2FmxStencil-js.html%23mxStencil)