# MoTiVML Guideline Document

This is a guideline document for the MoTiVML language. This guideline document assists end users in modelling variability in robotic systems. As a supporting artifact to the language, this document serves as a form of verification for the MoTiVML language and its capabilities.

---

## 1 Prerequisites

- Ubuntu 20.04.3 LTS and above
- ROS installation. This application was developed and tested with ROS Neotic 1.15.9.
- Python3 installation
- C++ 11 or higher

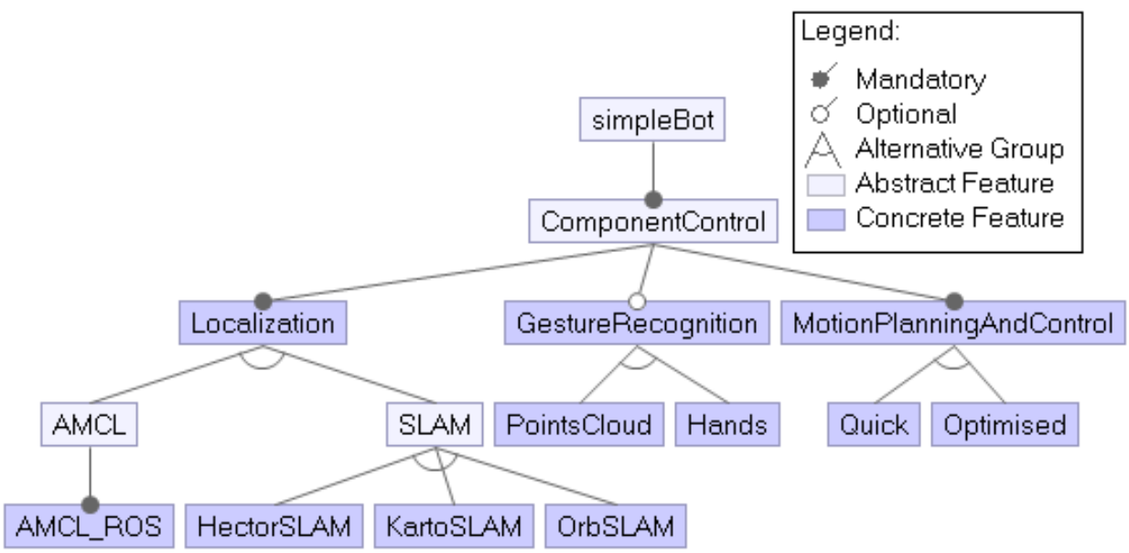
## 2 Setup and Installation of MoTiVML

- Clone MoTiVML from here into your catkin workspace.
- Go the MoTiVML workspace project directory and build the cloned project with the command catkin_make and press enter on your keyboard.

## 3 How To

This section gives a step-by-step walk-through of how MoTiVML can be used to instantiate a model or configuration, how it can be used to validate a model as well as the capability to link source code implementations with feature definitions. We also give a brief tutorial of our in-built MoTiVML console interface. Throughout our demonstrations in this guideline document, we use a sample model of a robotic system that is shown graphically in figure 1. This graphical representation of a simple robot model will also serve as our running example throughout this document.

Figure 1: Graphical Model of a Simple Robot




### 3.1 Instantiate a MoTiVML Model

- To create an instance of a model in MoTiVML, in the **src** folder of the cloned application, make a copy of the template folder provided and rename it to suit your preference.
- In your instance folder, there is a **model.json** with an existing root feature, wrapped in a object.

- The model definition can be extended by appending sub features to the root feature. You can add nested feature objects according to your preferences.Note that each feature specification must have attributes such as **id**, **name**, **constraints**, **group**, **isMandatory**. The constraints attribute must contain sub attributes **featureIncluded**, **featureExcluded**, **bindingTimeAllowed**, **bindingModeAllowed**. Listing 1 shows a demonstrated example of an extended MoTiVML model instance, for the feature model captured in Figure 1.

- The value of each feature attribute in a model must conform to strict type systems defined within MoTiVML. A list of all the attributes together with their expected values is provided below in Table 1.

Table 1: Feature Attributes and Types

| Attribute | Expected Value |
|---|---|
| id | Alphanumeric characters. No spaces allowed. |
| name | Alphanumeric characters. No spaces allowed. |
| constraints | Object containing attributes **featureIncluded**, **featureExcluded**, **bindingTimeAllowed**, **bindingModeAllowed** |
| featureIncluded | Array of existing feature "ids" |
| featureExcluded | Array of existing feature "ids" |
| bindingTimeAllowed | Early / Late / Any |
| bindingModeAllowed | Static / Dynamic / Any |
| group | XOR / OR |
| isMandatory | True or False |

- To add a sub-feature to a defined feature, add the **sub** attribute to the feature and assign an array of sub-feature objects to it. However, the sub attribute is optional. Features that do not have sub-features can exist without a **sub** attribute.

Listing 1: SimpleRobot Model Instantiation in MoTiVML

```
1
2  {
3          "id": "root_feature",
4          "name": "SimpleBot",
5          "group": "",
6          "isMandatory": true,
7          "isSelected": true,
8          "sub": [
9          {
10                 "id": "compcontrol",
11                 "name": "ComponentControl",
12                 "constraints": {
13                         "featuresIncluded": [],
14                         "featuresExcluded": [],
15                         "bindingTimeAllowed":"Early",
16                         "bindingModeAllowed":"Static"
17                 },
18                 "group": "OR",
19                 "isMandatory": true,
20                 "sub":[
21                 {
22                         "id": "localztn",
23                         "name": "Localisation",
24                         "constraints": {
25                                 "featuresIncluded": [],
26                                 "featuresExcluded": [],
27                                 "bindingTimeAllowed":"Early",
28                                 "bindingModeAllowed":"Static"
29                         },
30                         "group": "OR",
31                         "isMandatory": true,
32                         "sub":[
33                         {
34                                 "id": "amcl",
35                                 "name": "AMCL",
36                                 "constraints": {
37                                         "featuresIncluded": [],
38                                         "featuresExcluded": [],
39                                         "bindingTimeAllowed":"Early",
40                                         "bindingModeAllowed":"Static"
41                                 },
42                                 "group": "XOR",
43                                 "isMandatory": false,
```

```
                                "sub":[
                                {
                                        "id": "amclros",
                                        "name": "AmclRos",
                                        "constraints": {
                                                "featuresIncluded": [],
                                                "featuresExcluded": [],
                                                "bindingTimeAllowed":"Early",
                                                "bindingModeAllowed":"Static"
                                        },
                                        "group": "OR",
                                        "isMandatory": true
                                }
                                ]
                        },
                        {
                                "id": "slam",
                                "name": "SLAM",
                                "constraints": {
                                        "featuresIncluded": [],
                                        "featuresExcluded": [],
                                        "bindingTimeAllowed":"Early",
                                        "bindingModeAllowed":"Static"
                                },
                                "group": "XOR",
                                "isMandatory": false,
                                "sub":[
                                {
                                        "id": "hectorslam",
                                        "name": "HectorSLAM",
                                        "constraints": {
                                                "featuresIncluded": [],
                                                "featuresExcluded": ["kartoslam", "orbslam"],
                                                "bindingTimeAllowed":"Late",
                                                "bindingModeAllowed":"Dynamic"
                                        },
                                        "group": "XOR",
                                        "isMandatory": false
                                },
                                {
                                        "id": "kartoslam",
                                        "name": "KartoSLAM",
                                        "constraints": {
                                                "featuresIncluded": [],
                                                "featuresExcluded": ["orbslam", "hectorslam"],
                                                "bindingTimeAllowed":"Late",
                                                "bindingModeAllowed":"Dynamic"
                                        },
                                        "group": "XOR",
                                        "isMandatory": false
                                },
                                {
                                        "id": "orbslam",
                                        "name": "OrbSLAM",
                                        "constraints": {
                                                "featuresIncluded": [],
                                                "featuresExcluded": ["kartoslam", "hectorslam"],
                                                "bindingTimeAllowed":"Late",
                                                "bindingModeAllowed":"Dynamic"
                                        },
                                        "group": "XOR",
                                        "isMandatory": false
                                }
                                ]
                        }
                        ]
                },
                {
                        "id": "gestrec",
                        "name": "GestureRecognition",
                        "constraints": {
                                "featuresIncluded": [],
                                "featuresExcluded": [],
                                "bindingTimeAllowed":"Early",
                                "bindingModeAllowed":"Static"
```

```
                        },
                        "group": "OR",
                        "isMandatory": false,
                        "sub":[
                        {
                                "id": "hands",
                                "name": "Hands",
                                "constraints": {
                                        "featuresIncluded": [],
                                        "featuresExcluded": ["pointscloud"],
                                        "bindingTimeAllowed":"Late",
                                        "bindingModeAllowed":"Dynamic"
                                },
                                "group": "XOR",
                                "isMandatory": false
                        },
                        {
                                "id": "pointscloud",
                                "name": "PointsCloud",
                                "constraints": {
                                        "featuresIncluded": [],
                                        "featuresExcluded": ["hands"],
                                        "bindingTimeAllowed":"Late",
                                        "bindingModeAllowed":"Dynamic"
                                },
                                "group": "XOR",
                                "isMandatory": false
                        }
                        ]
                },
                {
                        "id": "motplannctrl",
                        "name": "MotionPlanningAndControl",
                        "constraints": {
                                "featuresIncluded": [],
                                "featuresExcluded": [],
                                "bindingTimeAllowed":"Early",
                                "bindingModeAllowed":"Static"
                        },
                        "group": "OR",
                        "isMandatory": true,
                        "sub":[
                        {
                                "id": "quick",
                                "name": "Quick",
                                "constraints": {
                                        "featuresIncluded": [],
                                        "featuresExcluded": ["optimised"],
                                        "bindingTimeAllowed":"Late",
                                        "bindingModeAllowed":"Dynamic"
                                },
                                "group": "XOR",
                                "isMandatory": false
                        },
                        {
                                "id": "optimised",
                                "name": "Optimised",
                                "constraints": {
                                        "featuresIncluded": [],
                                        "featuresExcluded": ["quick"],
                                        "bindingTimeAllowed":"Late",
                                        "bindingModeAllowed":"Dynamic"
                                },
                                "group": "XOR",
                                "isMandatory": false
                        }
                        ]
                }
                ]
        }
        ]
}

```

### 3.2 Instantiate a MoTiVML Configuration

- Again in your instance folder, there is a **config.json** with an empty **properties** array value.
- In the config.json file of your project, for every feature added in your model.json file, a corresponding configuration object must be added. For each feature configuration object, there must be an **id** attribute which references the **id** of a feature in your model.json file. The **props** attribute must contain **time** and **mode** attributes. Listing 2 shows a MoTiVML configuration translation of the features present in Figure 1.
- The values of each configuration attribute for your model must conform to strict type systems defined within MoTiVML. A list of all the configuration attributes together with their expected values is provided below in Table 2.

Table 2: Configuration Attributes and Types

| Attribute | Expected Value |
|---|---|
| id | Alphanumeric characters. No spaces allowed. |
| props | Object containing attributes **time** and **mode** |
| time | Early / Late |
| mode | Static / Dynamic |

Listing 2: SimpleRobot Model Configuration Instantiation in MoTiVML

```
1
2   {
3           "properties": [
4           {
5                   "id": "compcontrol",
6                   "props": {
7                           "time":"Early",
8                           "mode":"Static"
9                   }
10          },
11          {
12                  "id": "localztn",
13                  "props": {
14                          "time":"Early",
15                          "mode":"Static"
16                  }
17          },
18          {
19                  "id": "gestrec",
20                  "props": {
21                          "time":"Early",
22                          "mode":"Static"
23                  }
24          },
25          {
26                  "id": "motplannctrl",
27                  "props": {
28                          "time":"Early",
29                          "mode":"Static"
30                  }
31          },
32          {
33                  "id": "amcl",
34                  "props": {
35                          "time":"Early",
36                          "mode":"Static"
37                  }
38          },
39          {
40                  "id": "amclros",
41                  "props": {
42                          "time":"Early",
43                          "mode":"Static"
44                  }
45          },
46          {
47                  "id": "slam",
48                  "props": {
49                          "time":"Early",
50                          "mode":"Static"
51                  }
```

```
        },
        {
                "id": "hectorslam",
                "props": {
                        "time":"Late",
                        "mode":"Dynamic"
                }
        },
        {
                "id": "kartoslam",
                "props": {
                        "time":"Late",
                        "mode":"Dynamic"
                }
        },
        {
                "id": "orbslam",
                "props": {
                        "time":"Late",
                        "mode":"Dynamic"
                }
        },
        {
                "id": "hands",
                "props": {
                        "time":"late",
                        "mode":"Dynamic"
                }
        },
        {
                "id": "pointscloud",
                "props": {
                        "time":"Late",
                        "mode":"Dynamic"
                }
        },
        {
                "id": "quick",
                "props": {
                        "time":"Late",
                        "mode":"Dynamic"
                }
        },
        {
                "id": "optimised",
                "props": {
                        "time":"Late",
                        "mode":"Dynamic"
                }
        }
        ]
}
```

### 3.3 Validate a MoTiVML Model

To validate your model and its configuration, navigate to the **/src/motivml** directory and run the following command in the ROS console:

Listing 3: MoTiVML Validation Command

```
python motivml.py <project_directory_name>
```

All models created in the MoTiVML language are validated on two distinct levels. i.e. syntactically and semantically. Syntactical validation has to do with feature and configuration schemas being evaluated for errors, while semantic validation has to do with modelling and binding constraints evaluation.

To demonstrate this we evaluate our constructed model and configuration shown in Listing 1 and 2 and discuss the results accordingly. A careful observation of the console output generated from the validation indicates the presence of zero errors.

When our in-built schema and constraint checker identify an error in either a model or a configuration, an error message is outputted to the console indicating where the error is located and why the language has flagged it as an error.

Figure 2: Simplebot Validation Output

```
C:\catkin_ws\src\motivml_ros\src\motivml>python motivml.py simplebot
Compiling simplebot
Generating Early Bindings
-------------------
Reading configuration: C:\catkin_ws\src\motivml_ros\src\simplebot\config.json
--------------Parsing Model----------------
[SUCCESS] Model Schema Has Zero Syntax Errors
--------------Model Parsing Ended-----------

--------------Parsing Binding-----------------
[SUCCESS] Model Binding Schema Has Zero Syntax Errors
-------------- Binding Parsing Ended-----------

--------------Constraint Checker Validation-----------------
[SUCCESS] Constraint Checker Found Zero Constraint Errors
```

Upon successfully validating a model without any errors, a **bindings.motivml** file is generated in the corresponding project folder. This file contains features that are have been bound at compile time. At runtime, the **bindings.motivml** file is dynamically updated depending on which features are successfully bound or unbound.

### 3.4 Plug in Source Code Implementations of Model Features

In the event that there are no syntax and semantic errors detected by the MoTiVML engine, source code implementations of features can be implemented and executed. With regards to attaching feature implementations, MoTiVML is language agnostic. This means that, source code implementation in both C++ and Python can be plugged in successfully. A source code implementation can be plugged in through following the following steps:

- Extract, encapsulate and plug in your feature implementations into the featx directory
- For each plugged in source code implementation, name the entrypoint file containing the main function of your implementation, with the same name as the corresponding feature in your MoTiVML model.json file.
- In the MoTiVML command line interface, a number of commands can be executed to perform specific tasks.

### 3.5 MoTiVML Console Interface

To add some interactivity to our variability modelling language, we have provided an in-built console interface for interacting with models and features through MoTiVML specific commands.

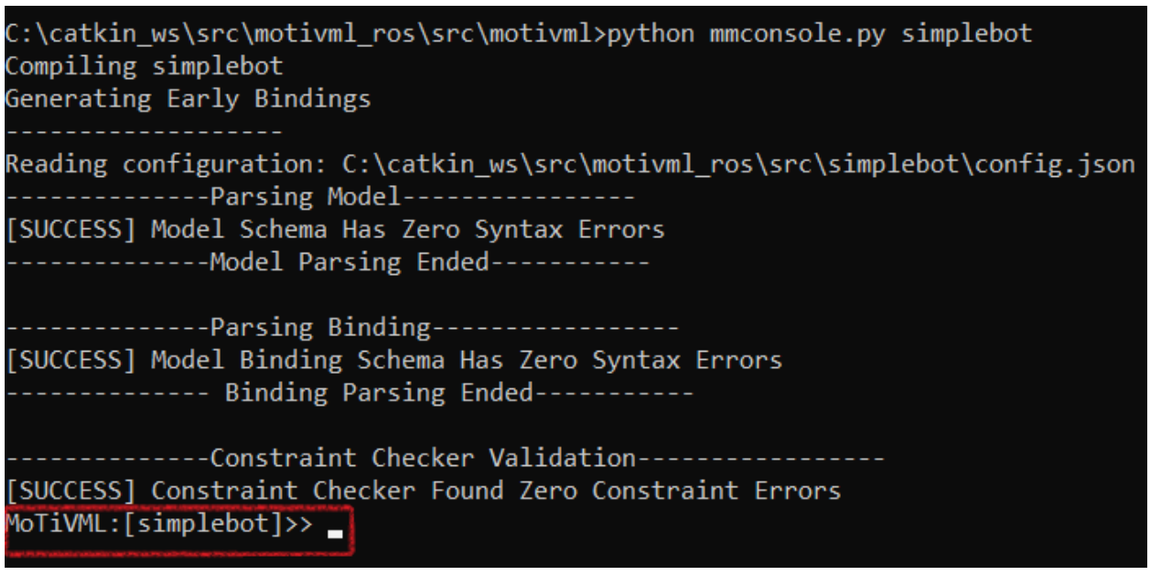
To launch the MoTiVML console interface, navigate to the **/src/motivml** directory and run the following command in Listing 4:

Listing 4: MoTiVML Console Launch Command

```
1
2 python mmconsole.py <project_directory_name>
3
```

When the above command is run, the model and configuration defined in the **project_directory_name** is compiled and validated, prior to the console interface being launched. If there are no errors detected, the console interface is then launched successfully.

Figure 3: Simplebot Console



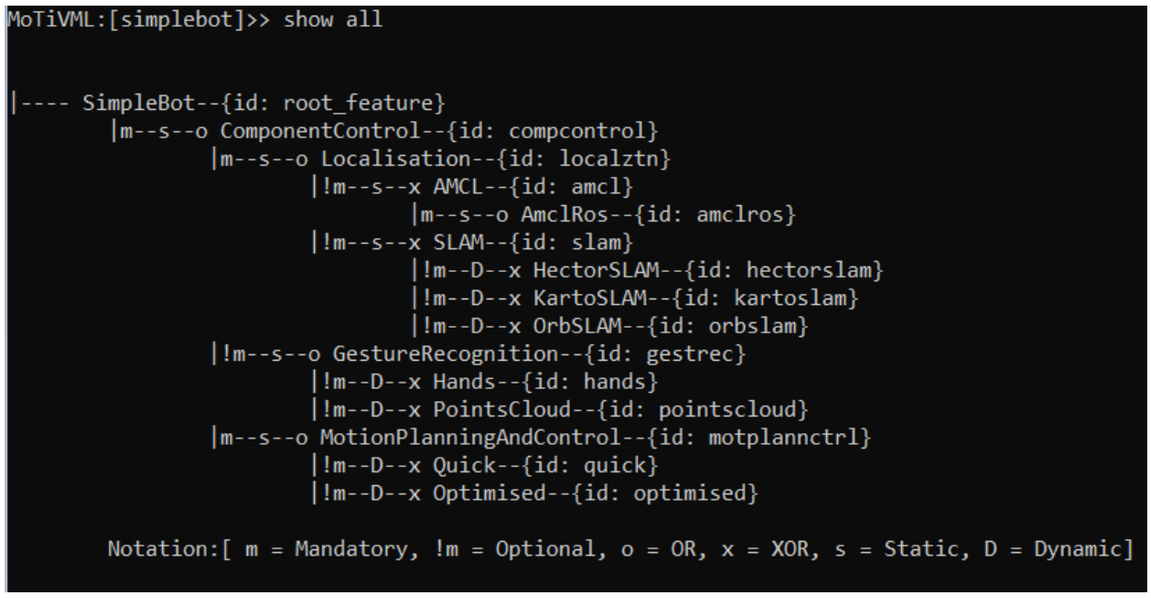

In Figure 3 we can observe the launched MoTiVML console. As highlighted in the figure above, the console prompt always bears the name of the project which was launched with the console. For this reason, all MoTiVML console commands that are run will be affiliated with the model highlighted in the console prompt.

## 3.6 MoTiVML Console Commands

- **show** < *parameter* >: The show command is used to display graphically a user constructed model. This command prints the hierarchical structure of a selected model while displaying details such as its mandatory status, group and binding mode. The show command takes a single parameter. This parameter can either be **all** or **config**.

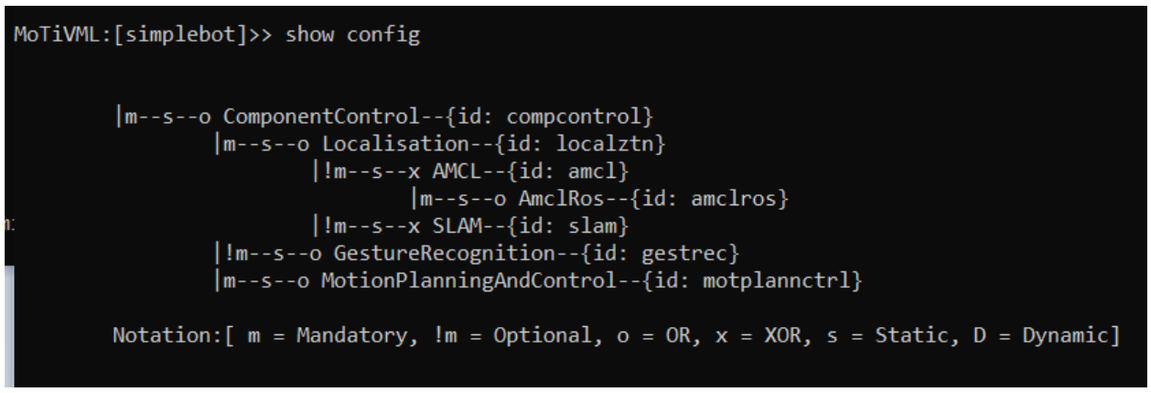
  The **show all** command only shows all selected features in the model tree. By default all features are selected when instantiated. Thus, this command shows every single feature in the model tree.

Figure 4: show all Command




The **show config** command only shows currently selected or bound features. That is, unselected or unbound features in the model are excluded from the output tree.

Figure 5: show config Command




- **run** $< feature\_name >$: The run command executes the source code implementation plugged into a model in connection to a defined feature. This means that before the run command can be executed successfully, there must be an existing source code implementation in either C++ or python that has been plugged into a model. The command only takes one parameter which is the name of the feature you wish run.

  C++ feature implementations require a compile step to be able to run them. That in the **src/motivml/featx/mbin** directory of our implementation, we save all C++ feature executables after compilation and then proceed to run them.

- **exit**: The exit command shuts down the console interface and returns back to the original ROS console. The exit command also asks for a yes or no confirmation before proceeding. Figure 6 shows an exit command execution and output.

Figure 6: Exit Command