TATA CONSULTANCY SERVICES
Experience certainty.

NCRA • TIFR

TATA

Extending Model-Driven Engineering in Tango

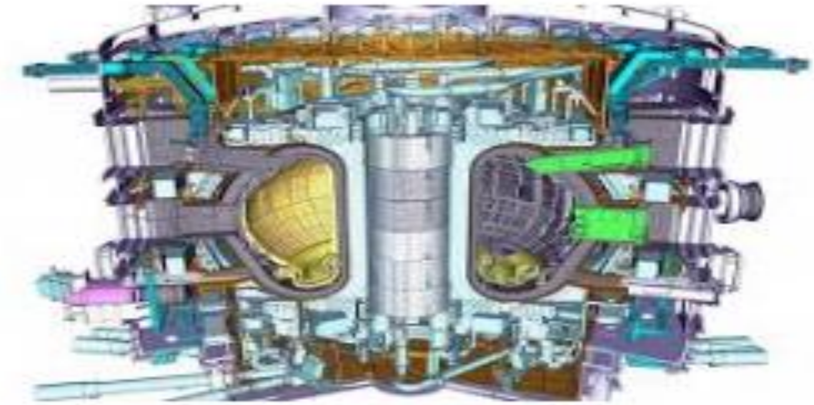
Amar Banerjee
Puneet Patwari
Subhrojyoti Roy Chaudhuri
Swaminathan Natarajan

Agenda

- Context & History
- Objectives: Drivers for the MDE Approach
- Approach Overview
  - Control Systems DSL
  - Code generation
  - Simulator Generation
  - Verification Support
  - Logging and Log Analysis
- Contribution Possibilities

Context & History

- 200+ plant systems: desire for standardized control system development using common platform and control architecture
- Specifications to cover all controls aspects: commands, alarms, data processing, system structure
- Major source of value: integrated model of the entire control system



ITER

SKA

GMRT

- New control systems solution for uGMRT
- Control systems DSL (domain-specific language)
- Facilitates design, test case generation, verification through simulation, logging and log analysis Tango code generation
- Collaboration with South Africa on simulation

- Leading Telescope Manager Consortium
- Tango-centric design solution
- Compatible with DSL approach

Objectives: Drivers for the MDE Approach

1. Integrated control system model
   - How control nodes (Tango device drivers) collaborate to achieve control system capabilities
   - Comprehensive interface specifications (ref. POGO) covering commands, responses, alarms, data
2. Domain-specific language to express control logic
   - State machine specification of control, integrated with command validation, data processing and alarm handling specifications
   - Semantic transparency of desired control systems behaviour, facilitating automation of verification and generation of simulators
   - Complex algorithms (e.g. dish pointing) and orchestration specifications can be handled through callouts, scripting plugins (or coded in DSL).
3. Extend approach to engineering life cycle
   - Enable logging, log analysis, simulator generation, verification support: enable complete life cycle at same abstraction level

Control Systems DSL: Concept

```
Model WeatherSimulatorDevice
InterfaceDescription WeatherSim_ID
{
    dataPoints{float Temperature [], float Insolation [],
               float Pressure [], float Rainfall [],
               float Wind_Speed [], float Wind_Direction [],
               float Relative_Humidity []}

    commands{ON[], OFF[], RESET[]}

    responses{RES_ON[string msg], RES_OFF[string msg], RES_RESET[string msg]}

    operatingStates{
        SWITCHED_ON[], SWITCHED_OFF[], ALARM[], STANDBY[]
        startState : SWITCHED_ON
        endState : SWITCHED_OFF
    }
}
ControlNode WeatherSimulator_CN
{
    Associated Interface Description : WeatherSim_ID

    DataPointBlock{DataPoint WeatherSimulatorDevice.WeatherSim_ID.Temperature{
                DataPointHandling{DataPointValidation[Max Value = 55 Min Value = -10]}}
```

```
CommandResponseBlock{Command WeatherSimulatorDevice.WeatherSim_ID.ON {
        Transitions{currentState WeatherSimulatorDevice.WeatherSim_ID.SWITCHED_OFF => nextState WeatherSimulatorDevice.WeatherSim_ID.SWITCHED_ON}
        ResponseBlock{expectedResponse WeatherSimulatorDevice.WeatherSim_ID.RES_ON {
        ResponseValidation {parameter WeatherSimulatorDevice.WeatherSim_ID.RES_OFF.msg []}}}}
```

Control Systems DSL – GMRT pilot example

Interface Description

```
Model GMRT
InterfaceDescription GMRT_ID{
    dataPoints {
        string tdbArchiver="archival/tooll/db"[
            string AttributeList="tango://01hw587782:10000/lmc/c01/ofcsnt/lt2pow",
            string DbHost="01hw587782",
            string DbPort="10000",
            string DbUser="root",
            string DbPassword="root"
        ],
        string alarmServer="alarmServer/test/1"[],
        string deviceName="GMRT/Servo/1"[],
        string subSystemId="SERVO"[],
        string deviceProperty_DisplayAt="StartUP"[]
        float dynamic_DP3[],
        float SERVO_WIN_VEL=0.0 [
            string comment="WIND VELOCITY SENSOR VA
            string AttributeIndex="666",
            string AttributeDataType="float",
            string DisplayAt="StartUp",
            string name = "SubstystemLaunched",
            boolean isPolled = true
        ],
        int SERVO_PWR_AC=0 [
            string comment="POWER SENSOR VALUE",
            string AttributeIndex="321",
            string AttributeDataType="float",
            string DisplayAt="StartUp"
        ],
        int subSystemLaunched[
            string name = "SubstystemLaunched",bool
        ],
        string response[
            int minValue=100,
            int maxAlarm = 200
        ],
```

```
alarms {
    SERVO_WIN_VEL_SENSR2_QUALITY[
        string AlarmList="ATTR_ALARM",
        string AlarmReceivers="SNAP",
        string AlarmDescription="SERVO WIND VELOCITY QUALITY",
        string AlarmSeverities="WARNING"
    ],
    SERVO_PWR_AC_EL_QUALITY[
        string AlarmList="ATTR_ALARM",
        string AlarmReceivers="SNAP",
        string AlarmDescription="SERVO POWER QUALITY",
        string AlarmSeverities="WARNING"
    ]

}
commands {
// The command specific details which go into the custom database are configure
    POSITION[

        string cmdHId ="555",
        string cmdUId ="867",
        string numHPkt ="111",
        string numUPkt ="222",
        string timeout ="900",
        string priority="0",
        string alias="POS",
        string hint="hint: position <Axis_1> <Angle>  <Axis_2> <Angle> <Ang
Angle = <deg>:<arc>:<sec> deg = -270 to 270 arc = 0 to 59 sec = 0 to 59"
```

Control Systems DSL – GMRT pilot example

Behavior Description

```
ControlNode GMRT_CN{
    Associated Interface Description : GMRT_ID

    // Define dynamic behavior of the alarm
    AlarmBlock {
        Alarm GMRT_ID.SERVO_PWR_AC_EL_QUALITY{

            // Specify the alarm triger conditions to
            AlarmTriggerCondition {

                DataPoints : GMRT.GMRT_ID.SERVO_PWR_A(
            }
            AlarmHandling {
                // Specify actions for alarms
                Action [
                    fireCommands : GMRT.GMRT_ID.STOP
                     // Specify the script you want tc
                Op  OP1 execute "File Path Of Script"
                ]
            }

        }
        Alarm GMRT.GMRT_ID.SERVO_WIN_VEL_SENSR2_QUALI1
            AlarmTriggerCondition {
                DataPoints : GMRT.GMRT_ID.SERVO_WIN_VI
            }

        }
```

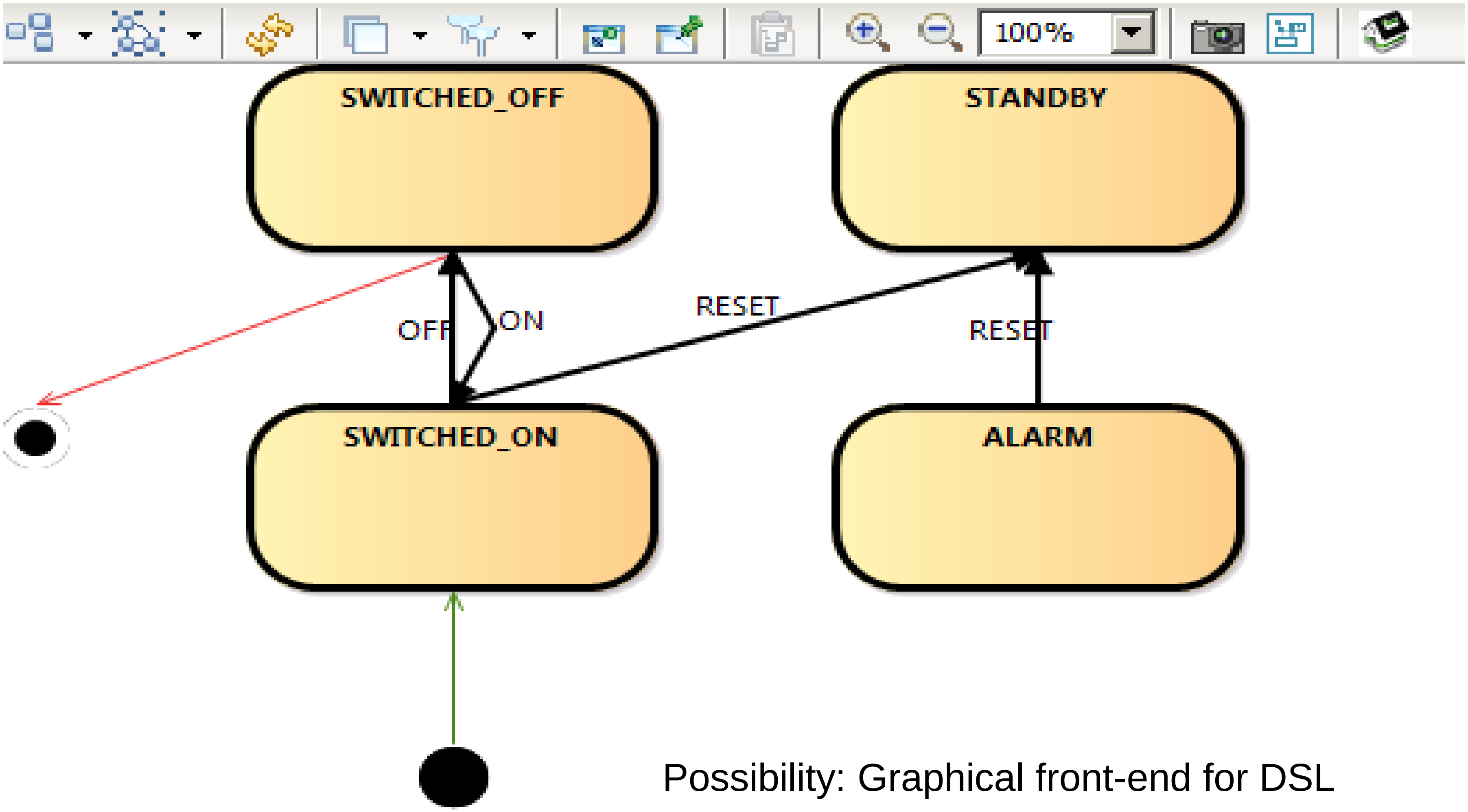
```
CommandResponseBlock {

    Command GMRT.GMRT_ID.HOLD {
        CommandValidation {
            parameter GMRT.GMRT_ID.HOLD.para1 [
                Min Value = 0
                Max Value = 200
                Possible Values = (20,50,60,70)
            ]
        }
        Transitions {

            currentState GMRT.GMRT_ID.operationalManual (exitAction  Action [])
            => nextState GMRT.GMRT_ID.operationalAutomatic


        }
    }
    Command GMRT.GMRT_ID.POSITION {
        CommandValidation {

            parameter GMRT.GMRT_ID.POSITION.para1 [
                Min Value = 0
                Max Value = 200
                Possible Values = (20,50,60,70)
            ]
        }
        Transitions {
            currentState GMRT.GMRT_ID.initialization
            => nextState GMRT.GMRT_ID.operationalManual
        }

    }
```

Consistency Check Across Devices

```
TangoSimLib for WeatherSimulator_CN {
    dataSimulations {
Couldn't resolve reference to Attribute 'WeatherSimulator_CN.Temperatur'.
                data_Simulation_Algorithm ConstantQuantity{
                    initialValue 44.0
                    quality 3.0
                }
            },
            Attribute WeatherSimulator_CN.Insolation {
                data_Simulation_Algorithm
            },
            Attribute WeatherSimulator_CN.Pressure {
                data_Simulation_Algorithm
            },
            Attribute WeatherSimulator_CN.Rainfall {
                data_Simulation_Algorithm
            },
            Attribute WeatherSimulator_CN.Wind_Speed {
                data_Simulation_Algorithm
            },
            Attribute WeatherSimulator_CN.Wind_Direction {
                data_Simulation_Algorithm
            },
            Attribute WeatherSimulator_CN.Relative_Humidity {
                data_Simulation_Algorithm
            }
    }
```

Generated Graphical View

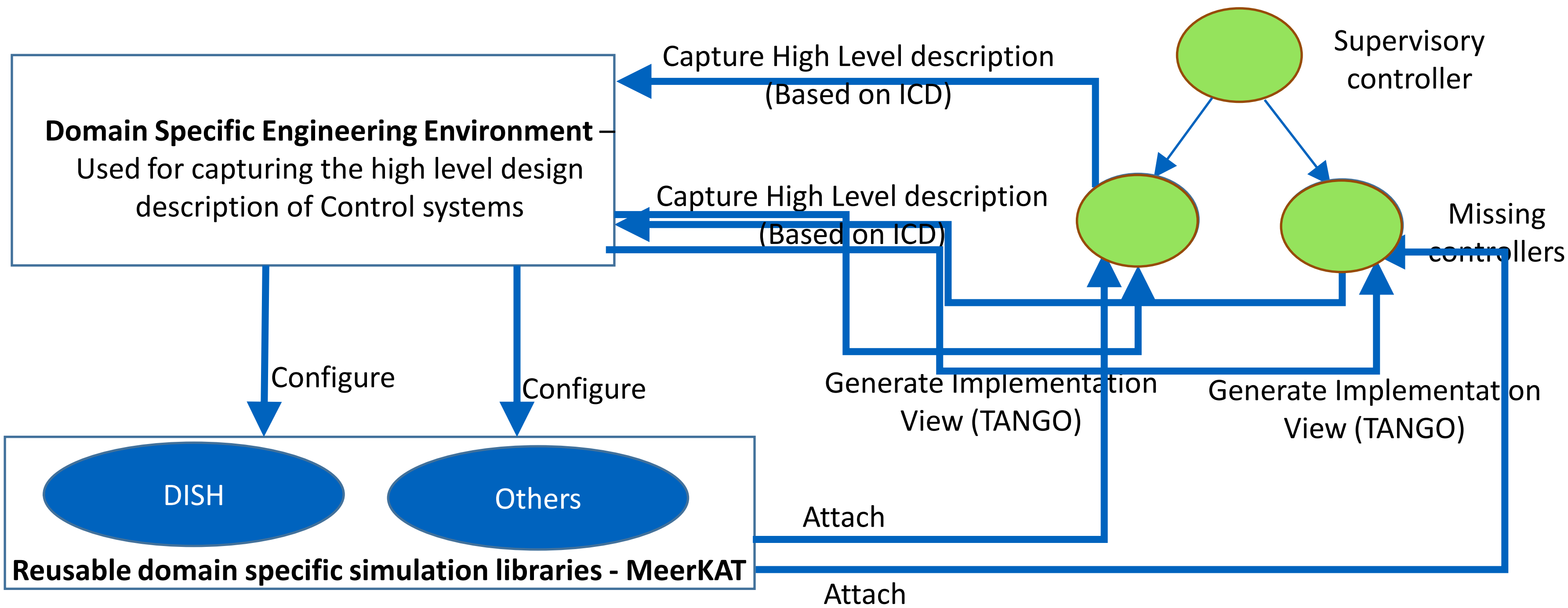
Early Prototype



Possibility: Graphical front-end for DSL

Simulation Support – Indo-SA prototype

- **Approach that reduces significantly the effort to implement simulators for various subsystem controllers**



MeerKAT – Weather Simulator

Capture description using DSL

```
Model WeatherSimulatorDevice
InterfaceDescription WeatherSim_ID
{
    dataPoints{float Temperature [], float Insolation [],
               float Pressure [], float Rainfall [],
               float Wind_Speed [], float Wind_Direction [],
               float Relative_Humidity []}

    commands{ON[], OFF[], RESET[]}

    responses{RES_ON[string msg], RES_OFF[string msg], RES_RESET[string msg]}

    operatingStates{
        SWITCHED_ON[], SWITCHED_OFF[], ALARM[], STANDBY[]
        startState : SWITCHED_ON
        endState : SWITCHED_OFF
    }
}
ControlNode WeatherSimulator_CN
{
    Associated Interface Description : WeatherSim_ID

    DataPointBlock{DataPoint WeatherSimulatorDevice.WeatherSim_ID.Temperature{
                DataPointHandling{DataPointValidation[Max Value = 55 Min Value = -10]}}
```

```
CommandResponseBlock{Command WeatherSimulatorDevice.WeatherSim_ID.ON {
        Transitions{currentState WeatherSimulatorDevice.WeatherSim_ID.SWITCHED_OFF => nextState WeatherSimulatorDevice.WeatherSim_ID.SWITCHED_ON}
        ResponseBlock{expectedResponse WeatherSimulatorDevice.WeatherSim_ID.RES_ON {
        ResponseValidation {parameter WeatherSimulatorDevice.WeatherSim_ID.RES_OFF.msg []}}}}
```

Simulator Generated Code

```
InterfaceDescription ID{
    commands{
        c1[], c2[]
    }
    dataPoints{
        float d1[], float d2[]
    }
    operatingStates{
        s1[], s2[]
    }
}

ControlNode CN{
    Associated Interface Description : ID
    CommandResponseBlock {
        Command test.ID.c1{
            Transitions {
                currentState test.ID.s1 => nextState test.ID.s2

            }
        }
    }
}
```

Capture of controller description (M&C ML)

Auto - Generate

Controller (TANGO Device)

Test Case

```
commands {
    Command c1 {
        description c1 argin Argument {
            type VoidType
        }
        argout Argument {
            type VoidType
        }
        status InheritanceStatus {
            inherited ^false concrete ^true concre
        }
    }
}
attributes {
    Attribute d1 {
        attType Scalar rwType READ dataType Double
            inherited ^false concrete ^true concre
        }
        properties AttrProperties {
            description "" label "" unit "" standa
            maxValue "0" minValue "0" maxWarning "
            deltaValue ""
        }
    },
    Attribute d2 {
        attType Scalar rwType READ dataType DoubleType status InheritanceStatus
            inherited ^false concrete ^true concreteHere ^true
        }
        properties AttrProperties {
            description "" label "" unit "" standardUnit "" displayUnit "" forma
            maxValue "0" minValue "0" maxWarning "" minWarning "" deltaTime ""
            deltaValue ""
        }
    }
}
```

Controller view in TANGO (POGO)

SimLib configuration view

```
TangoSimLib for CN {
    dataSimulations {
        Attribute CN.d1 {
            data_Simulation_Algorithm
        },
        Attribute CN.d2 {
            data_Simulation_Algorithm
        }
    }
    behaviours {
        Command CN.c1 {
        },
    }
}
```

Auto - Generate

Controller (TANGO Device)

Auto - Generate

SimLib JSON

Generation of Test Cases

- The DSL includes the control state machine of the device
  - We can use this to generate Junit test cases (really controller devices and stub device simulators) that exercise the state machine
  - Start it in an initial state, issue commands with parameters that satisfy validation constraints, check whether the target device state changes as expected
  - Can exercise various combinations of legal paths
  - Can also exercise illegal commands that do not pass validation tests
- DSL also includes alarm detection logic
  - Can supply data values intended to trigger alarms and monitor the resulting behaviour
- Similarly stub simulators can be programmed to generate valid and invalid data values
- Need human-specified configuration files or annotations to identify more sophisticated test cases

Test Cases - Example

Can derive the test cases from the design – Initial prototype

```
@DataProvider(name="on")
    public Object[][] onDataProvider() {
    return new Object[][]{


            new Object[] {"{\"fixedResponse\":{\"Response\":\"RES_ON\",\"msg\":0},\"ON\":[]}"},

            new Object[] {"{\"fixedResponse\":{\"Response\":\"RES_ON\",\"msg\":0},\"ON\":[]}"},

            new Object[] {"{\"ON\":[]}"},

            new Object[] {"{\"ON\":[]}"},

            new Object[] {"{\"ON\":[]}"},


                        };
}

@Test(dataProvider="off")
        public void OFF(String params) throws DevFailed {
        DeviceProxy dp =new  DeviceProxy("nodes/WeatherSimulator_CN/test");
                        DeviceData dd = new fr.esrf.TangoApi.DeviceData();
    dd.insert(params);
    String resp = dp.command_inout("OFF",dd).extractString();
    System.out.println(resp);
    Assert.assertEquals(resp, "RES_OFF:-msg:0||");
```

Generated POGO View

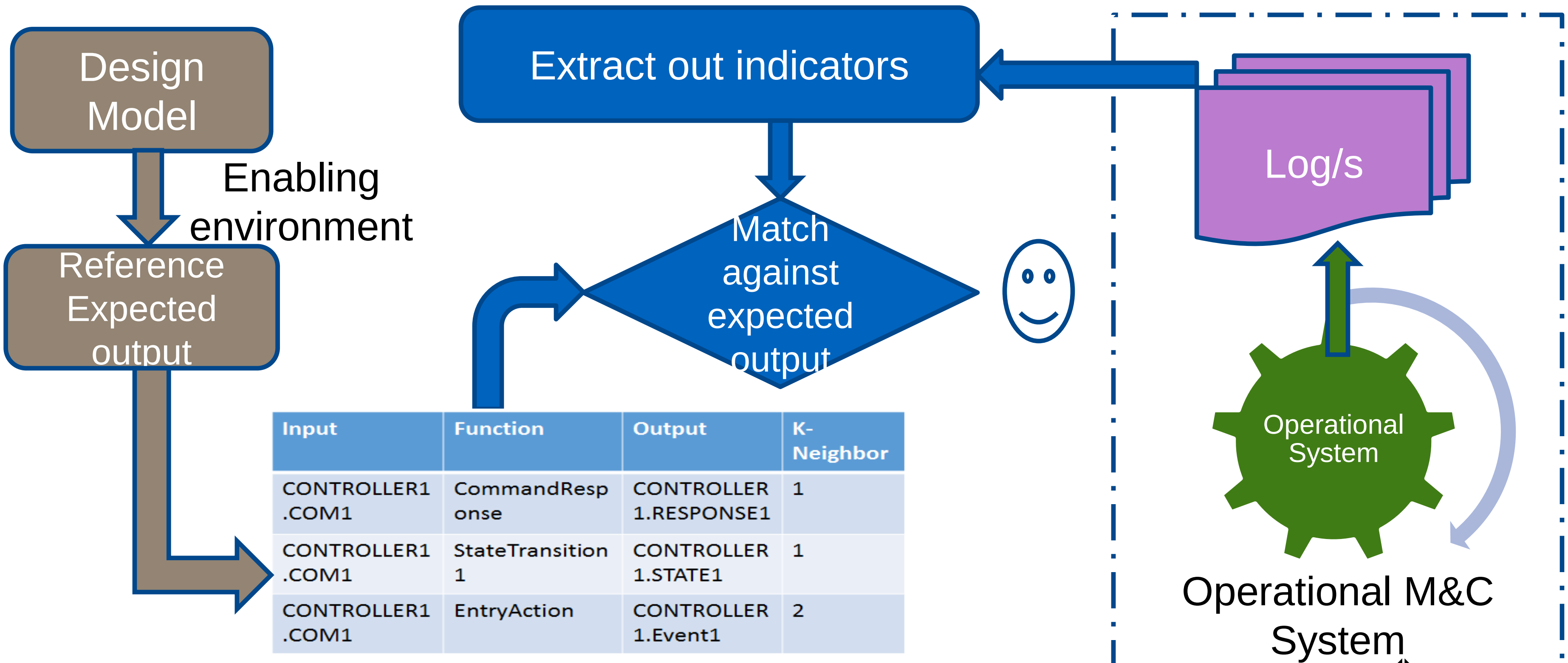
```
	Command OFF {
		description OFF argin Argument {
			type VoidType
		}
		argout Argument {
			type VoidType
		}
		status InheritanceSta
			inherited ^false
		}
	},
	Command RESET {
		description RESET arg
			type VoidType
		}
		argout Argument {
			type VoidType
		}
		status InheritanceSta
			inherited ^false
		}
	}
}
attributes {
	Attribute Temperature {
		attType Scalar rwType READ dataType DoubleType status InheritanceStatus {
			inherited ^false concrete ^true concreteHere ^true
		}
```

```java
package com.mnc.pogo.nodes.java;

/*----- PROTECTED REGION ID(WeatherSimulator_CN.imports) ENABLED START -----*/
import org.slf4j.Logger;

/*----- PROTECTED REGION END -----*/	//	WeatherSimulator_CN.imports

/**
 *  WeatherSimulator_CN class description:
 *    WeatherSimulator_CN
 */

@Device
public class WeatherSimulator_CN {

	protected static final Logger logger = LoggerFactory.getLogger(WeatherSimulator_CN.class);
	protected static final XLogger xlogger = XLoggerFactory.getXLogger(WeatherSimulator_CN.class);
	//========================================================
	//  Programmer's data members
	//========================================================
	/*----- PROTECTED REGION ID(WeatherSimulator_CN.variables) ENABLED START -----*/

	//	Put static variables here
```

Log File Analysis

- Ensure robust design, strong traceability between design, its realization and operations for minimum control system downtime.
- DSL make the semantics of desired behaviour visible e.g. command responses, alarms and their handling, commands to be sent etc
  - Can automatically generate logging for the significant activities, and also automatic log analysis to check the actual behaviour against expected

DSL



Input	Function	Output	K-Neighbor
CONTROLLER1 .COM1	CommandResp onse	CONTROLLER 1.RESPONSE1	1
CONTROLLER1 .COM1	StateTransition 1	CONTROLLER 1.STATE1	1
CONTROLLER1 .COM1	EntryAction	CONTROLLER 1.Event1	2

Generated Java Code in TANGO – Logging Support

```
public PipeValue getMnc_ALARM() {
    xlogger.entry();
    // Write programmer code
    xlogger.exit();
    return this.Mnc_ALARM;
}

public void setMnc_ALARM(PipeValue Mnc_ALARM) throws DevFailed {
    xlogger.entry();
    deviceManager.pushPipeEvent("Mnc_ALARM", Mnc_ALARM);
    setStatus("Alarm Raised :: "+Mnc_ALARM.getValue().getName());
    this.Mnc_ALARM = Mnc_ALARM;
    xlogger.info("ALARM::"+Mnc_ALARM.getValue().getName());
    xlogger.exit();
}

@Pipe(name = "Mnc_STATE", label = "STATE", displayLevel = DispLevel._OPERATOR)
private PipeValue Mnc_STATE;
public PipeValue getMnc_STATE() {
    xlogger.entry();
    // Write programmer code
    xlogger.info("CURRENT_STATE::"+Mnc_STATE.getValue().getName());
    xlogger.exit();
    return Mnc_STATE;
}
public void setMnc_STATE(PipeValue STATE) {
    xlogger.entry();
    xlogger.info("NEXT_STATE::"+STATE.getValue().getName());
    Mnc_STATE = STATE;
    xlogger.exit();
}
```

Summary

- Control Systems DSL adds a layer over Tango that captures both the interface and behavioural specification of devices
  - Can be thought of as an extension of POGO that includes behavioural logic specifications as well
- DSL specifications for different nodes integrate to capture the architecture of the control system
  - How devices collaborate to achieve different aspects of control
- This visibility to the control systems functional concept facilitates simulation, verification and log analysis
  - Also visualization of the control system design

Contribution Possibilities

- India (NCRA-TIFR, working in collaboration with industry partners such as TCS) could contribute these MDE capabilities developed for GMRT to Tango Controls, if there is interest
  - A layer over the Tango Platform, like POGO
    - Can even be thought of as a next generation POGO
    - Along with engineering life cycle support
- Open to a dialogue with members of the Tango community
  - Aligning the contribution
  - Contribution logistics

GMRT

SKA SQUARE KILOMETRE ARRAY
TELESCOPE MANAGER

TATA CONSULTANCY SERVICES
Experience certainty.

Questions ???

For more information, contact:
N. Swaminathan
Swami.n@tcs.com

NCRA • TIFR