TATA CONSULTANCY SERVICES
Experience certainty.

TOWARDS BUILDING REUSABILITY IN CONTROL SYSTEMS – A JOURNEY

ICALEPCS 15, Melbourne, Australia

Subhrojyoti C*, Puneet Patwari, Amar, Banerjee, N. Swaminathan, G Muralikrishna,

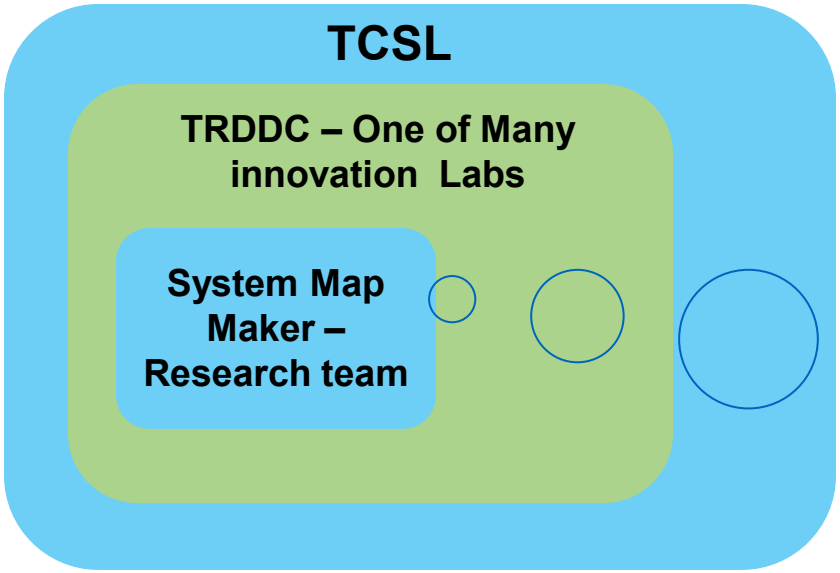
Tata Research Development and Design Centre, Tata Consultancy Services,

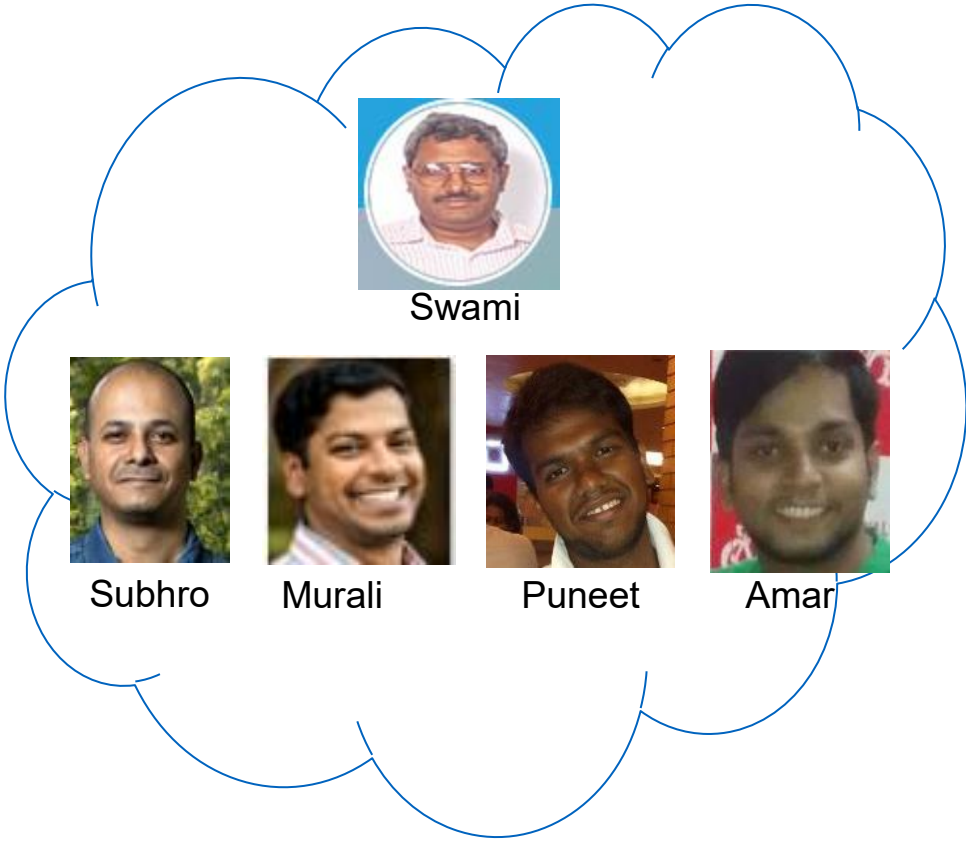
subhrojyoti.c@tcs.com, +91-20-66086411

Talk - Overview

- Brief about who we are
- Challenges and opportunities
  - Towards reusability
- Specific problem areas
  - Reuse across all phases of development lifecycle
  - Solution approach – MDE approach with a focus on knowledge reuse
- Conclusion
  - Where are we going with this – towards knowledge based systems

Who we are – System Map Maker

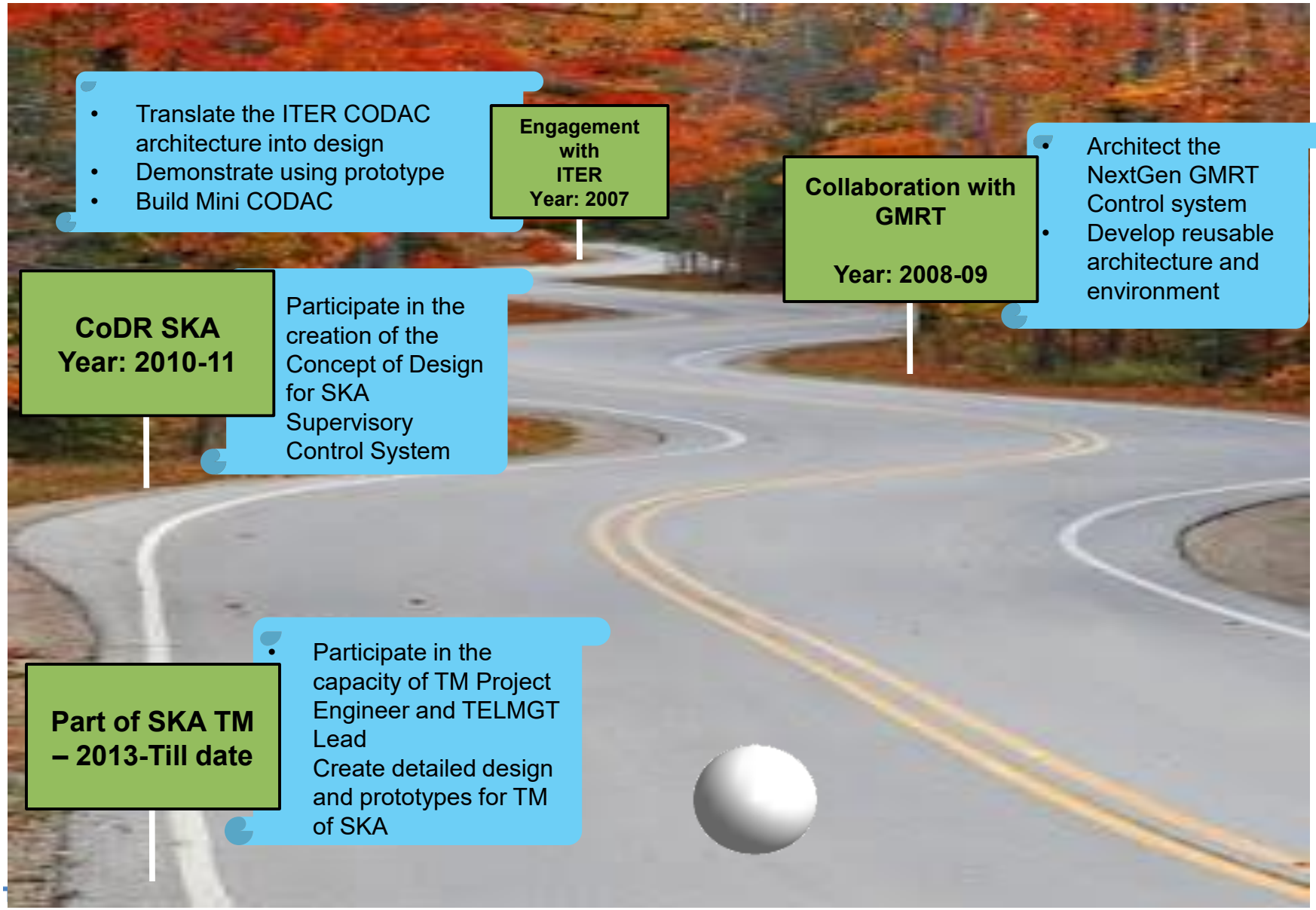




Broad area of focus:

- Integrated family of models that capture all the systems engineering information related to a particular system
  - Work out information architecture to facilitate System Engineering

Collaborations - Control systems



Key observation – Opportunity/Motivation

- Very robust solutions available for reuse for the purpose of implementation
  - EPICS, TANGO, ACS and so on

- Challenges and areas of opportunities

  - Significant amount of time spent in the early design life cycle
    - E.g. – ~ 3 years to design TM

  - Each project recreate the design thinking from scratch – and reinvent common M&C concerns, requirements, solution concepts and so on.
    - Usage of SysML does help communicate better, but still falls short on the reusability aspect !

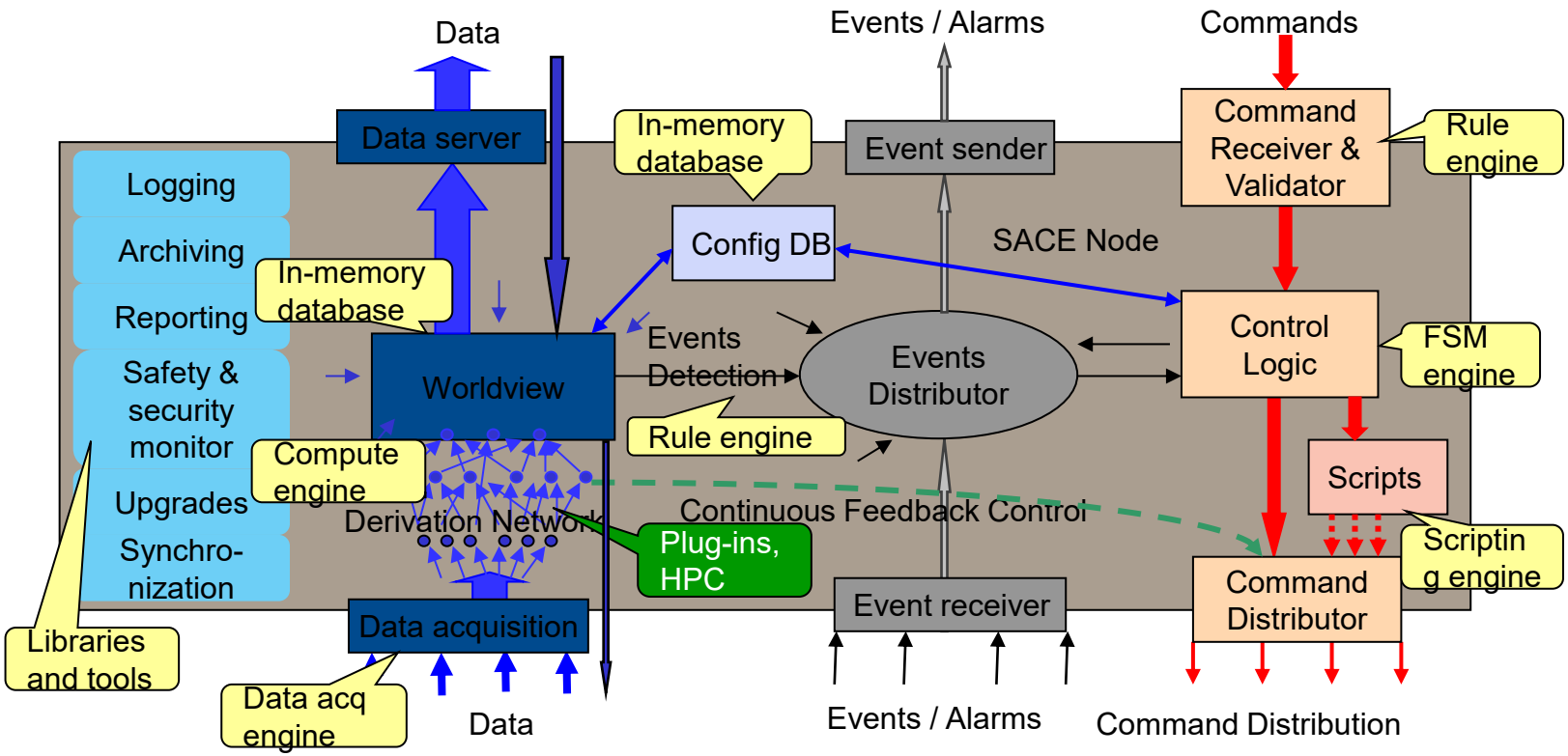
  - Much of the knowledge reside with the domain experts

Solution opportunity – Start with reusable architecture M&C

Can we create a common solution architecture for M&C - requirements, non functional concerns, functional blocks and so on

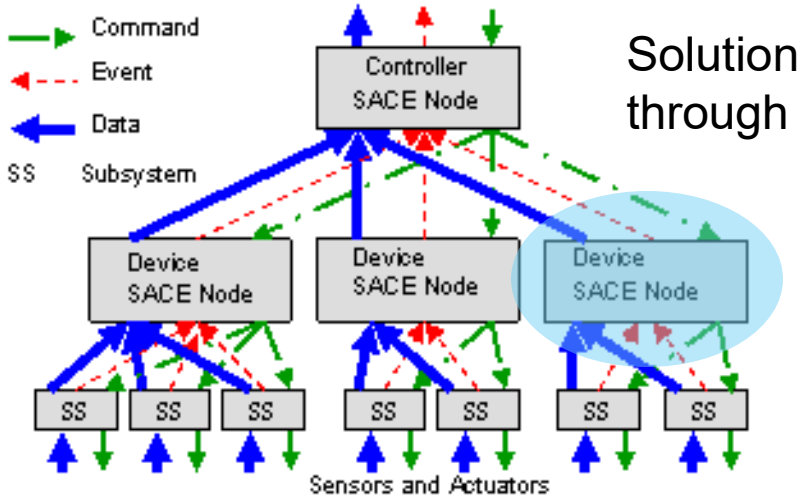
Solution opportunity – Start with reusable architecture M&C

Can we create a common solution architecture for M&C - requirements, non functional concerns, functional blocks and so on



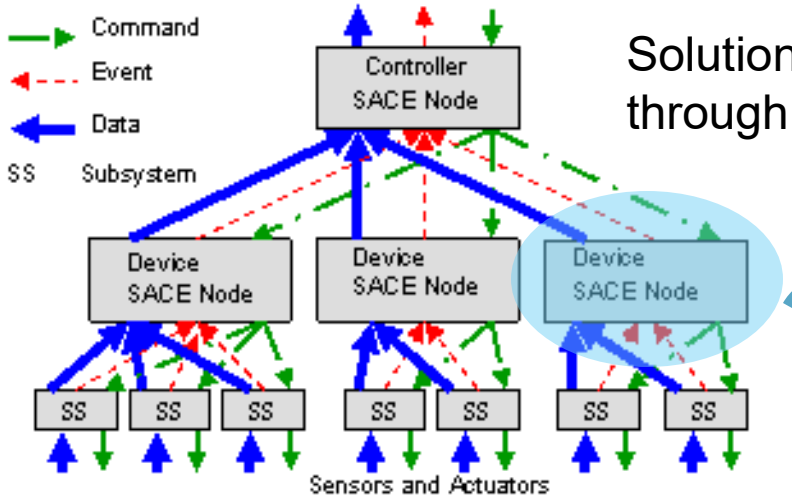
Independent of implementation technology,
Retargetable to multiple off the shelf options
Completely Data Driven

Identification of the input data



Solution to each control problem created through the composition of such nodes

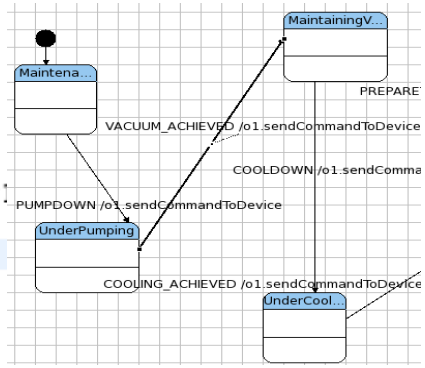
Identification of the input data



Solution to each control problem created through the composition of such nodes

Each node configured using its Self Description

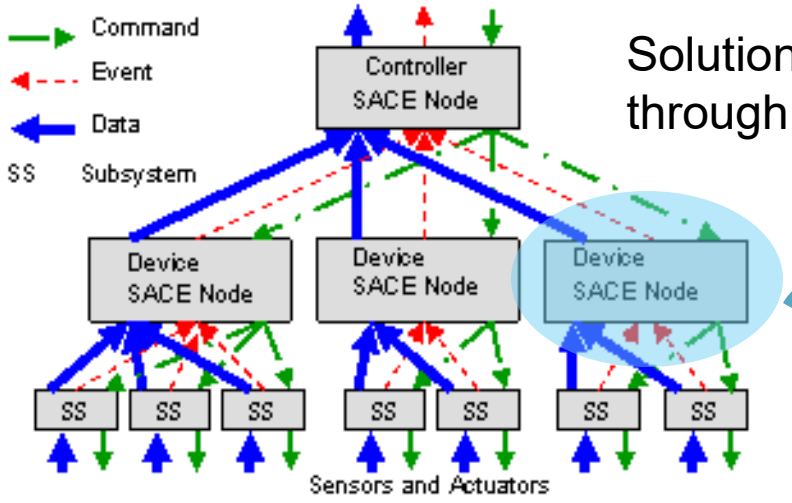
```
<Command Name = "PUMPDOWN"/>
<Command Name = "COOLDOWN"/>
<Command Name = "PREPARETF">
    <CommandAttachment Name = "PowerSupply1
</Command>
<Signal Name = "Cryo_Vacuum_Pressure" Type
    <UpdateRate Value = "10"/>
</Signal>
```



Action Name	Parameter	Values
PUMPDOWN	DataSim:Amplitude	60
	DataSim:Offset	40
	DataSim:Frequency	1
	DataSim:Sine	1
	DataSim:Square	-1
	DataSim:Triangle	
	DataSim:UpdateFrequency	
	DataSim:Ramp	
COOLDOWN	DataSim:Amplitude	15
	DataSim:Offset	7
	DataSim:Frequency	1
	DataSim:Sine	1
	DataSim:Square	-1
	DataSim:Triangle	

Device Name	Stream Name	Byte at location	Has values	Event
MagnetSystem	Cryo_Vacuum_Pressure		24	VACUUM_ACHIEVED
MagnetSystem	TF_Temperature		20	QUENCH_OCCURRED
MagnetSystem	Cryo_Vacuum_Pressure		0	COOLING_ACHIEVED

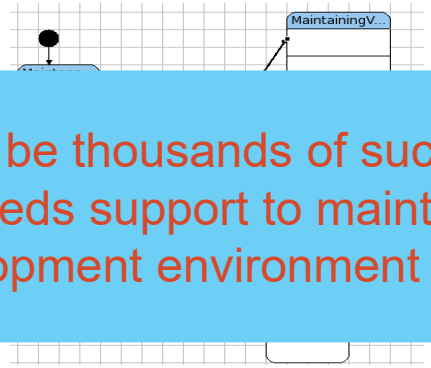
Identification of the input data



Solution to each control problem created through the composition of such nodes

Each node configured using its Self Description

```
<Command Name = "Pl
<Command Name = "C
<Command Name = "Pl
    <CommandAttachr
</Command>
<Signal Name = "Cr
    <UpdateRate Va
</Signal>
```



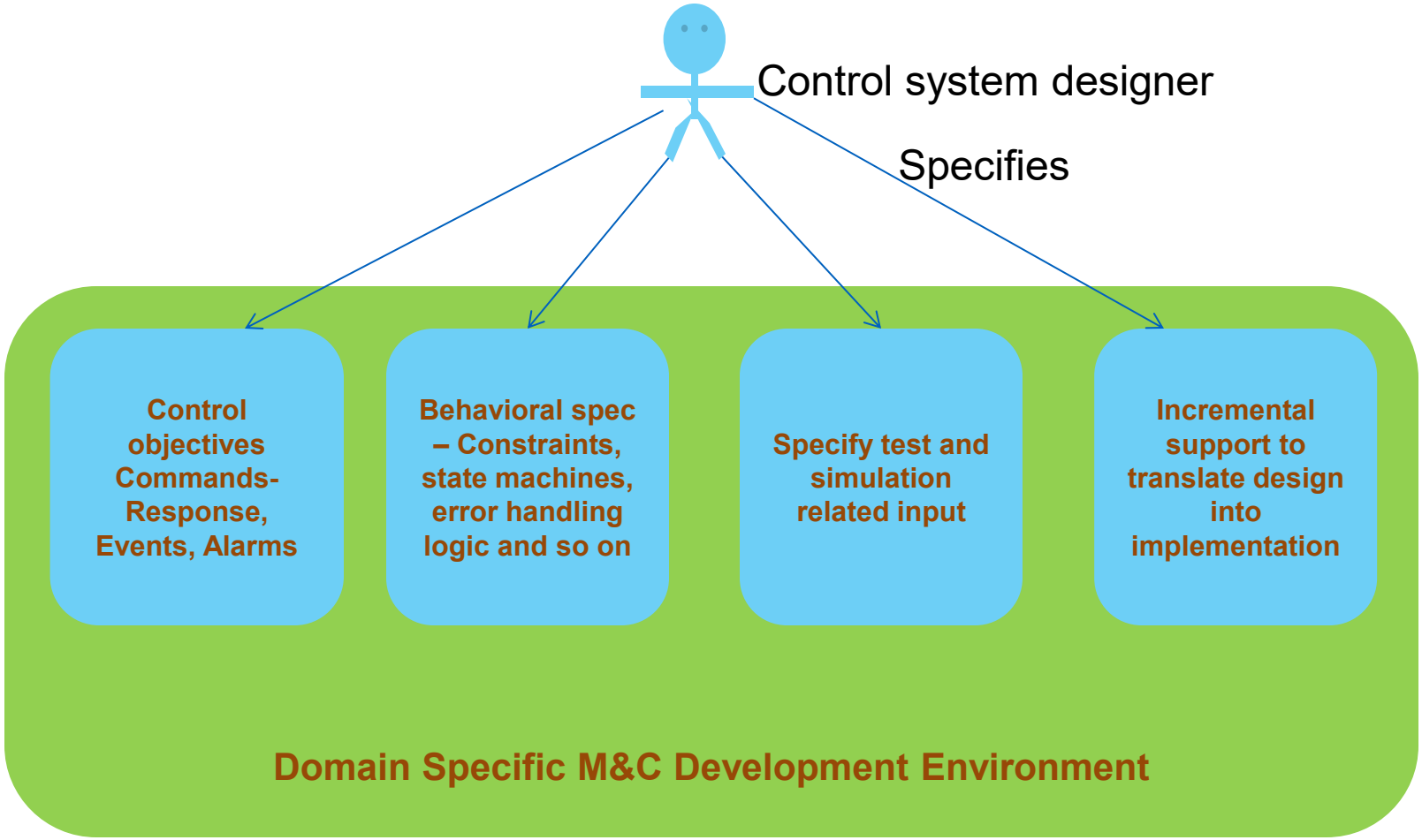
There might be thousands of such specifications and hence needs support to maintain consistency - A development environment in its own right!!

Action Name	Parameter	Values
	DataSim:Amplitude	60
		40
		1
		1
		-1
		15
	DataSim:Offset	7
	DataSim:Frequency	1
	DataSim:Sine	1
	DataSim:Square	-1
	DataSim:Triangle	

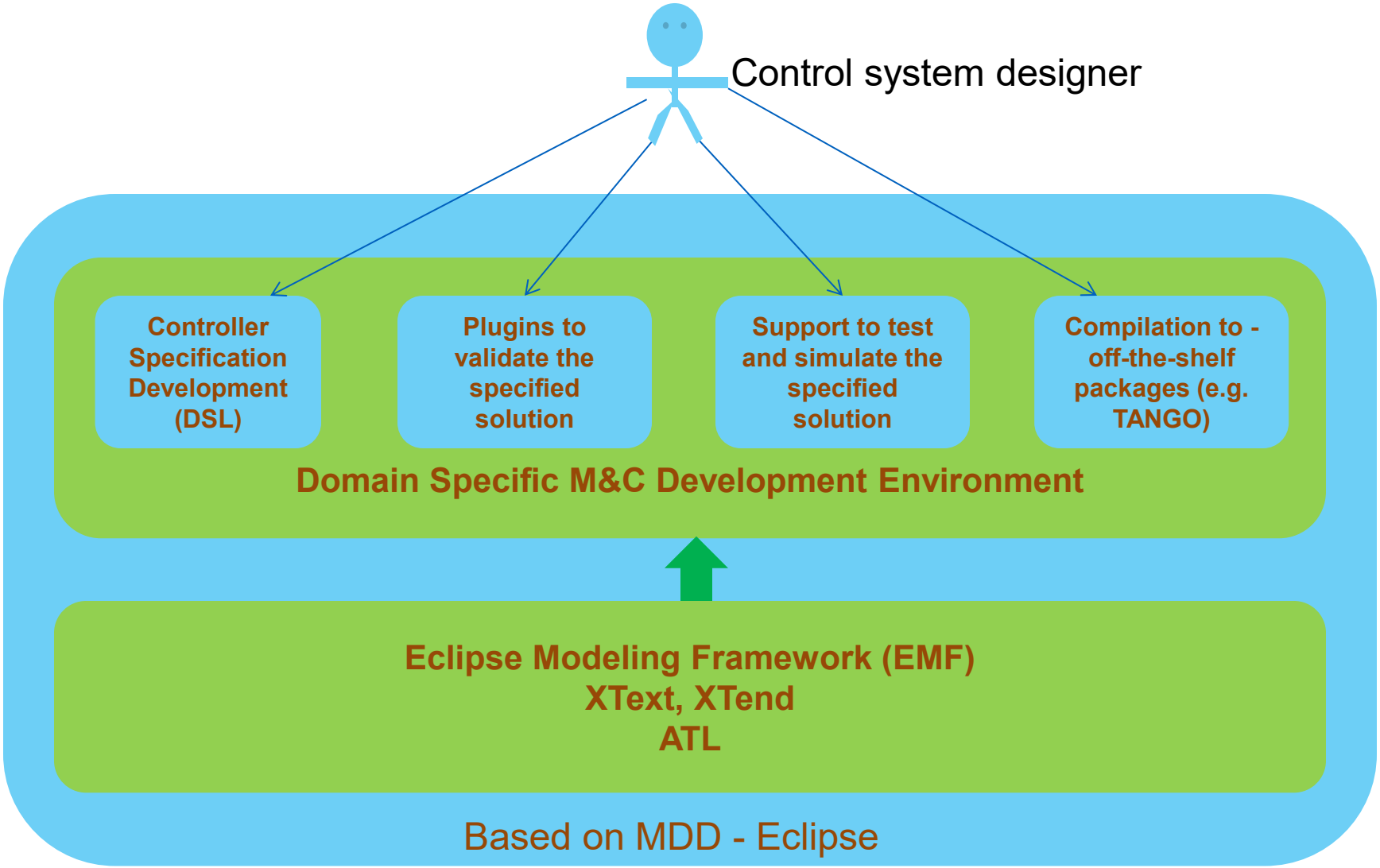
Device Name	Stream Name	Byte at location	Has values	Event
MagnetSystem	Cryo_Vacuum_Pressure		24	VACUUM_ACHIEVED
MagnetSystem	TF_Temperature		20	QUENCH_OCCURRED
MagnetSystem	Cryo_Vacuum_Pressure		0	COOLING_ACHIEVED

COOLDOWN

Architecting Environment – Domain Driven Engineering



M&C Architecting Environment –Implementation



Areas covered in the DSL as of now

- Specification of control structures
  - Control nodes
  - Commands, parameters
  - Responses
  - Events, Alarms
  - Data streams
  - States
- Behavior
  - Dependencies-Control Hierarchies
  - State machines
  - Command validation logic
  - Alarm detection logic
  - Alarm handling specifications
  - Data processing
- Placeholder
  - External/detailed logic

```
InterfaceDescription ID_Test{

    commands{
        Start[

        ]
    }
    dataPoints {
        int   a
    }

    events{
        event1[]
    }
    alarms{
        A1[]
    }
}
```

```
ControlNode AggregationNode{
    Associated Interface Description : GMRT.Controller_ID

    childNodes (GMRT.Controller)
    commandValidation {
        command GMRT.Controller_ID.DOSET
        [parameter bw1 (Possible Values = (6, 16, 32))]
    }
    statemachine AgnStMc{
        state GMRT.Controller_ID.Start
            [
                transitions
                event GMRT.Controller_ID.BOOT_Performed => state GMRT.Controller_ID.Connec
            ]
    }

}
```

alarmConditions
alarmHandling
commandTranslation
dataPointHandling
eventTranslation
operations
parentNode
responseValidation

Process – Working of the environment

```
Model GMRT
InterfaceDescription ID_IF{
    dataPoints{
        float IF_healthStatus = 10.0
    }
    commands{
        DOSET[
            parameter int bw1 = 1
            parameter int bw2 = 16
            parameter int alc1 = 0
            parameter int alc2 = 0
        ]
    }
    responses{
        RES_DOSET[
            parameter int DeviceNo = 10
            parameter int CmdStat = 0
            parameter string SETStr = ""
        ]
    }
    events{
        DOSET_Performed[]
    }
    states{
        Start[]
        Subscribe[]
    }
    commandResponseMap{
        command DOSET => expectedResponse RES_DOSET
    }
    commandEventMap{
        command DOSET => event DOSET_Performed
    }
    alarms{
        A1[ level : 5]
    }
}
```

```
ControlNode IF_Controller{
    Associated Interface Description : ID_IF
    statemachine IF_SM{
        state GMRT.ID_IF.Start[
            transitions
            event GMRT.ID_IF.DOSET_Performed => state GMRT.ID_IF.Subscribe
        ]
    }
    commandValidation{
        command GMRT.ID_IF.DOSET [
            parameter bw1 (Possible Values = (8, 16, 32)
            Max Value = 32 Min Value = 0)
            parameter bw2 (Possible Values = (8, 16, 32)
            Max Value = 32 Min Value = 0)
        ]
    }
    alarmConditions{
        AC2[
            dataPoint IF_healthStatus ( Min Value =5 )
            fireAlarm => GMRT.ID_IF.A1
        ]
    }
}
```

```
A1 - GMRT.ID_IF.A1
]
```

Specify M&C solution description using DSL – M&C ML

Process – Working of the environment

```
Model GMRT
InterfaceDescription ID_IF{
    dataPoints{
        float IF_healthStatus = 10.0
    }
    commands{
        DOSET[
            parameter int bw1 = 1
            parameter int bw2 = 16
            parameter int alc1 = 0
            parameter int alc2 = 0
        ]
    }
    responses{
        RES_DOSET[
            parameter int DeviceNo = 10
            parameter int CmdStat = 0
            parameter string SETStr = ""
        ]
    }
    events{
        DOSET_Performed[]
    }
    states{
        Start[]
        Subscribe[]
    }
    commandResponseMap{
        command DOSET => expectedResponse RES_DOSET
    }
    commandEventMap{
        command DOSET => event DOSET_Performed
    }
    alarms{
        A1[ level : 5]
```

```
ControlNode IF_Controller{
    Associated Interface Description : ID_IF
    statemachine IF_SM{
        state GMRT.ID_IF.Start[
            transitions
            event GMRT.ID_IF.DOSET_Performed => state GMRT.ID_IF.Subscribe
        ]
    }
    commandValidation{
        command GMRT.ID_IF.DOSET [
            parameter bw1 (Possible Values = (8, 16, 32)
            Max Value = 32 Min Value = 0)
            parameter bw2 (Possible Values = (8, 16, 3
            Max Value = 32 Min Value = 0)
        ]
    }
    alarmConditions{
        AC2[
            dataPoint IF_healthStatus ( Min Value =5 )
            fireAlarm => GMRT.ID_IF.A1
        ]
    }
}
```

```
A1 - GMRT.ID_IF.A1
]
```

```
}
```

Specify M&C solution description using DSL – M&C ML

Parse into

AST – Meta Model

- System
- ControlNode -> System
- InterfaceDescription -> System
- AbstractInterfaceItems
- AbstractOperationableItems
- OperatingState
- CommandResponseBlock
- Command -> AbstractInterfaceItems
- CommandValidation
- CommandDistribution
- CommandTranslation
- ParameterTranslation
- Transition
- Response -> AbstractInterfaceItems
- ResponseValidation
- ResponseDistribution
- ResponseTranslation
- ResponseTranslationRule
- Port
- EventBlock
- Event -> AbstractInterfaceItems, SubscribableItems
- EventCondition
- EventTranslation
- EventTranslationRule
- EventHandling
- AlarmBlock
- Alarm -> AbstractInterfaceItems, SubscribableItems
- AlarmCondition
- AlarmTranslation
- AlarmTranslationRule
- AlarmHandling
- DataPointBlock
- DataPoint -> AbstractInterfaceItems, SubscribableItems
- DataPointCondition

Process – Working of the environment

```
Model GMRT
InterfaceDescription ID_IF{
    dataPoints{
        float IF_healthStatus = 10.0
    }
    commands{
        DOSET[
            parameter int bw1 = 1
            parameter int bw2 = 16
            parameter int alc1 = 0
            parameter int alc2 = 0
        ]
    }
    responses{
        RES_DOSET[
            parameter int DeviceNo = 10
            parameter int CmdStat = 0
            parameter string SETStr = ""
        ]
    }
    events{
        DOSET_Performed[]
    }
    states{
        Start[]
        Subscribe[]
    }
    commandResponseMap{
        command DOSET => expectedResponse RES_DOSET
    }
    commandEventMap{
        command DOSET => event DOSET_Performed
    }
    alarms{
        A1[ level : 5]
```

```
ControlNode IF_Controller{
    Associated Interface Description : ID_IF
    statemachine IF_SM{
        state GMRT.ID_IF.Start[
            transitions
            event GMRT.ID_IF.DOSET_Performed => state GMRT.ID_IF.Subscribe
        ]
    }
    commandValidation{
        command GMRT.ID_IF.DOSET [
            parameter bw1 (Possible Values = (8, 16, 32)
            Max Value = 32 Min Value = 0)
            parameter bw2 (Possible Values = (8, 16, 3
            Max Value = 32 Min Value = 0)
        ]
    }
    alarmConditions{
        AC2[
            dataPoint IF_healthStatus ( Min Value =5 )
            fireAlarm => GMRT.ID_IF.A1
        ]
    }
}
```

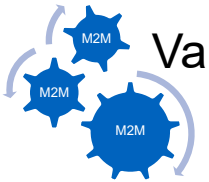
A1 - GMRT.ID_IF.A1

Specify M&C solution description using DSL – M&C ML

Parse into

AST – Meta Model

- System
- ControlNode -> System
- InterfaceDescription -> System
- AbstractInterfaceItems
- AbstractOperationableItems
- OperatingState
- CommandResponseBlock
- Command -> AbstractInterfaceItems
- CommandValidation
- CommandDistribution
- CommandTranslation
- ParameterTranslation
- Transition
- Response -> AbstractInterfaceItems
- ResponseValidation
- ResponseDistribution
- ResponseTranslation
- ResponseTranslationRule
- Port
- EventBlock
- Event -> AbstractInterfaceItems, SubscribableItems
- EventCondition
- EventTranslation
- EventTranslationRule
- EventHandling
- AlarmBlock
- Alarm -> AbstractInterfaceItems, SubscribableItems
- AlarmCondition
- AlarmTranslation
- AlarmTranslationRule
- AlarmHandling
- DataPointBlock
- DataPoint -> AbstractInterfaceItems, SubscribableItems
- DataPointCondition



Validation, M2M/T Translation…

Process – Working of the environment

```
Model GMRT
InterfaceDescription ID_IF{
    dataPoints{
        float IF_healthStatus = 10.0
    }
    commands{
        DOSET[
            parameter int bw1 = 1
            parameter int bw2 = 16
            parameter int alc1 = 0
            parameter int alc2 = 0
        ]
    }
    responses{
        RES_DOSET[
            parameter int DeviceNo = 10
            parameter int CmdStat = 0
            parameter string SETStr = ""
        ]
    }
    events{
        DOSET_Performed[]
    }
    states{
        Start[]
        Subscribe[]
    }
    commandResponseMap{
        command DOSET => expectedResponse RES_DOSET
    }
    commandEventMap{
        command DOSET => event DOSET_Performed
    }
    alarms{
        A1[ level : 5]
    }
}

ControlNode IF_Controller{
    Associated Interface Description : ID_IF
    statemachine IF_SM{
        state GMRT.ID_IF.Start[
            transitions
            event GMRT.ID_IF.DOSET_Performed => state GMRT.ID_IF.Subscribe
        ]
    }
    commandValidation{
        command GMRT.ID_IF.DOSET [
            parameter bw1 (Possible Values = {8, 16, 32)
            Max Value = 32 Min Value = 0)
            parameter bw2 (Possible Values = {8, 16, 3
            Max Value = 32 Min Value = 0)
        ]
    }
    alarmConditions{
        AC2[
            dataPoint IF_healthStatus ( Min Value =5 )
            fireAlarm => GMRT.ID_IF.A1
        ]
    }
}
```

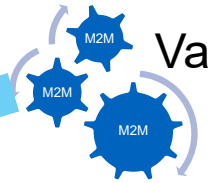
A1 - GMRT.ID_IF.A1

Specify M&C solution description using DSL – M&C ML

Parse into

AST – Meta Model

- System
- ControlNode -> System
- InterfaceDescription -> System
- AbstractInterfaceItems
- AbstractOperationableItems
- OperatingState
- CommandResponseBlock
- Command -> AbstractInterfaceItems
- CommandValidation
- CommandDistribution
- CommandTranslation
- ParameterTranslation
- Transition
- Response -> AbstractInterfaceItems
- ResponseValidation
- ResponseDistribution
- ResponseTranslation
- ResponseTranslationRule
- Port
- EventBlock
- Event -> AbstractInterfaceItems, SubscribableItems
- EventCondition
- EventTranslation
- EventTranslationRule
- EventHandling
- AlarmBlock
- Alarm -> AbstractInterfaceItems, SubscribableItems
- AlarmCondition
- AlarmTranslation
- AlarmTranslationRule
- AlarmHandling
- DataPointBlock
- DataPoint -> AbstractInterfaceItems, SubscribableItems
- DataPointCondition



Validation, M2M/T Translation…

Translation into TANGO implementation

```
@Device
public class Controller {
    private static final Logger logger = LoggerFactory.getLogger(Controller.class);
    private static final XLogger xlogger = XLoggerFactory.getXLogger(Controller.class);
    private String className = this.getClass().getName();
    private static String deviceName = "nodes/Controller/test";
    private String parentDeviceName = "nodes/LMC/test";
    private String[] childDeviceName = {};
    /*Declaire Variables Here */

    /* Variables Declaration Ends */

     /*----------------Initialization Code----------

     @Init(lazyLoading = false)
     public final void initDevice() throws DevFailed{
        xlogger.entry();
        logger.debug("init");
@Command(name = "DOSET")
public synchronized  String DOSET(String parameters) throws DevFailed {
    System.out.println("Executing command DOSET ([mncModel.impl.SimpleTypeImpl@73a460 (name: bw1
    String inCommand = new CommandHandler().getCurrentMethod(this);
    if (new NodeCommandResponseValidation().validateCommand(inCommand,parameters)) {
String responseReceived = new ControllerSimulator().simulateResponse(inCommand, parameters);
    // Code To Be Written
    if (new NodeCommandResponseValidation().validateResponse(inCommand,responseReceived)) {
        // Calling EventDOSET_Triggered();
            return responseReceived;
    }else{
        return new String("BAD RESPONSE");
    }
    } else {
        return new String("BAD COMMAND");
    }
}
```

Process – Working of the environment

```
Model GMRT
InterfaceDescription ID_IF{
    dataPoints{
        float IF_healthStatus = 10.0
    }
    commands{
        DOSET[
            parameter int bw1 = 1
            parameter int bw2 = 16
            parameter int alc1 = 0
            parameter int alc2 = 0
        ]
    }
    responses{
        RES_DOSET[
            parameter int DeviceNo = 10
            parameter int CmdStat = 0
            parameter string SETStr = ""
        ]
    }
    events{
        DOSET_Performed[]
    }
    states{
        Start[]
        Subscribe[]
    }
    commandResponseMap{
        command DOSET => expectedResponse RES_DOSET
    }
    commandEventMap{
        command DOSET => event DOSET_Performed
    }
    alarms{
        A1[ level : 5]
    }
}
```

```
ControlNode IF_Controller{
    Associated Interface Description : ID_IF
    statemachine IF_SM{
        state GMRT.ID_IF.Start[
            transitions
            event GMRT.ID_IF.DOSET_Performed => state GMRT.ID_IF.Subscribe
        ]
    }
    commandValidation{
        command GMRT.ID_IF.DOSET [
            parameter bw1 (Possible Values = {8, 16, 32}
            Max Value = 32 Min Value = 0)
            parameter bw2 (Possible Values = {8, 16, 3
            Max Value = 32 Min Value = 0)
        ]
    }
    alarmConditions{
        AC2[
            dataPoint IF_healthStatus ( Min Value =5 )
            fireAlarm => GMRT.ID_IF.A1
        ]
    }
}
```

A1 - GMRT.ID_IF.A1

Specify M&C solution description using DSL – M&C ML

Parse into

AST – Meta Model

- System
- ControlNode -> System
- InterfaceDescription -> System
- AbstractInterfaceItems
- AbstractOperationableItems
- OperatingState
- CommandResponseBlock
- Command -> AbstractInterfaceItems
- CommandValidation
- CommandDistribution
- CommandTranslation
- ParameterTranslation
- Transition
- Response -> AbstractInterfaceItems
- ResponseValidation
- ResponseDistribution
- ResponseTranslation
- ResponseTranslationRule
- Port
- EventBlock
- Event -> AbstractInterfaceItems, SubscribableItems
- EventCondition
- EventTranslation
- EventTranslationRule
- EventHandling
- AlarmBlock
- Alarm -> AbstractInterfaceItems, SubscribableItems
- AlarmCondition
- AlarmTranslation
- AlarmTranslationRule
- AlarmHandling
- DataPointBlock
- DataPoint -> AbstractInterfaceItems, SubscribableItems
- DataPointCondition

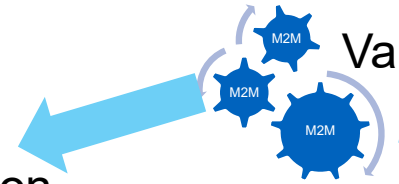
```
@Device
public class Controller {
    private static final Logger logger = LoggerFactory.getLogger(Controller.class);
    private static final XLogger xlogger = XLoggerFactory.getXLogger(Controller.class);
    private String className = this.getClass().getName();
    private static String deviceName = "nodes/Controller/test";
    private String parentDeviceName = "nodes/LMC/test";
    private String[] childDeviceName = {};
    /*Declaire Variables Here */

    /* Variables Declaration Ends */

     /*----------------Initialization Code----------

     @Init(lazyLoading = false)
     public final void initDevice() throws DevFailed{
        xlogger.entry();
        logger.debug("init");
@Command(name = "DOSET")
public synchronized  String DOSET(String parameters) throws DevFailed {
    System.out.println("Executing command DOSET ([mncModel.impl.SimpleTypeImpl@73a460 (name: bw1
    String inCommand = new CommandHandler().getCurrentMethod(this);
    if (new NodeCommandResponseValidation().validateCommand(inCommand,parameters)) {
String responseReceived = new ControllerSimulator().simulateResponse(inCommand, parameters);
    // Code To Be Written
    if (new NodeCommandResponseValidation().validateResponse(inCommand,responseReceived)) {
        // Calling EventDOSET_Triggered();
            return responseReceived;
    }else{
        return new String("BAD RESPONSE");
    }
    } else {
        return new String("BAD COMMAND");
    }
}
```

Translation into TANGO implementation



Validation, M2M/T Translation…
Generation of Simulators and Test Cases

```
public class ControllerSimulator {
    String parseResponse = "{\"DOMON\":[{\"RES_DOMON\":[{\"allowedValues\"
        String parseDataPoint = "{\"IF_Monitoring_HealthStatus\":{\"allowe
        String deviceName = "nodes/Controller/test";
        JSONObject dataPointJsonObject = (JSONObject) JSONValue.parse(pars
        JSONObject jsonkk = (JSONObject) JSONValue.parse(parseResponse);
Object object[] = dataPointJsonObject.keySet().toArray();
public String simulateResponse(String commandName, String commandParam
    JSONArray jsonArr = (JSONArray) jsonkk.get(commandName);
    if(jsonArr==null)
    {
        return "RESPONSE RECEIVED FOR "+commandName.toUpperCase();
    }
    JSONObject mlk = (JSONObject) jsonArr.get(0);

    JSONObject jso =  (JSONObject)JSONValue.parse(commandParameters);
    if (jso != null) {
    JSONObject fixedResp = (JSONObject) jso.get("fixedResponse");
    if(fixedResp!=null)
    {
    Set<String> map =  fixedResp.keySet();
    Iterator<String> iterat = map.iterator();
    String hh = fixedResp.get("Response")+":-";
    while(iterat.hasNext())
    {
        String mm = (String) iterat.next();
        if(!mm.equals("Response"))
        {
        hh= hh+ mm+":"+fixedResp.get(mm)+"||";
        }
    }
    System.out.println(hh);
    return hh;
```

Factors influencing DSL design

Key drivers to the DSL design

- Important to understand the stakeholder concerns
  - In our case M&C designers
  - Leave out the details of the application domain (e.g. Astronomy), projects specifics
- Understanding and extracting key concerns from the domain
  - Important input is the vocabulary using while specifying M&C design
- Capturing the underlying architecture pattern
  - SACE suggests a strict hierarchical
  - Hence the flow of the DSL reflects the same
  - However, cant be applied to other patterns, such as Agent based systems
- Decoupling the meta-modeling from the grammar definition
  - Capture the domain concerns in a structured form with relations
  - Don’t worry about the language syntax while doing so
- Careful analysis of the user intuitions for language syntax definition
  - Can be based on the combination of functional building blocks and architecture flows
- Extensibility
  - Support for incremental domain specific validations
  - Incremental addition of concepts (e.g. Radio telescope) and separation of concerns

Future direction – knowledge based engineering environment and conclusion

- We plan to try this our for uGMRT, SKA

- Such an environment is a stepping stone towards identification of domain models and capturing of knowledge –
  - Technical domain such as M&C, Networks, Security and so on
  - Application domain such as Radio astronomy Nuclear fusion

- Learnings can be used to build similar domain environment or contribute towards general capability such as SysML and so on.

Thank You!