Methodology to Develop Domain Specific Modeling Languages

Subhrojyoti Roy Chaudhuri
Swaminathan Natarajan
Amar Banerjee
(TCS Research and Innovation, Pune, Chennai)

Venkatesh Choppella
(IIIT, Hyderabad)
Talk outline

- Background story
- Technical description
- Discussions (Applications) & Roadmap
Background
ITER (International Thermonuclear Experimental Reactor)

- 200+ subsystems
- >100K sensors & actuators
- 3 years design time
- ~40 engineers for control systems

SKA (Square Kilometer Array)

- 1000’s of antennas
- 5 years design time
- 40+ engineers for control systems
Experience certainty.

**Challenges**

**Design**
- Highly effort intensive
- Past experience and domain knowledge dependent
- Involves multiple domains
- Geographically distributed teams.
- Numerous design decisions.

**Implementation**
- Multiple implementation technologies and protocols.
- Integration nightmare
- Larger testing efforts through commissioning.

**Evolution**
- Reconfiguration for operational efficiency.
- Technology adaptation, situations etc.
Typical practice

- Interface Control Documents (ICD)
  - A means to agree on the control responsibilities, e.g. Command validation, translation, state machine, event/alarm handling
  - Scattered across documents

- Use of SysML to model the complete system
  - Rarely used to develop an integrated model to generate code (e.g. EPICS, TANGO popular in the community)
  - Lack of support for control systems domain (vocabulary, design pattern, validation rules)

- Develop one off solutions
  - Implement one-off tools (E.g. SDD editor of ITER) for integration
  - Custom UI, custom model schema formats (in XML etc).
  - Difficult to reuse
Control systems features and characteristics

Significant similarity in the control systems functionality, features, architecture and solution patterns

Perfect candidate to be supported with a Domain Specific Modeling Language
Experience certainty.

Variabilities across Cyber-physical Systems: custom DSL for each domain?

Fusion Reactor  Radio Telescopes  Manufacturing Process Control  SmartGrid, Smart Home  Robotics and Smart Machines

Each domain might also decide to build its own DSL. Hence a **standard methodology** to develop DSLs will be of great value.
But how to approach?

A systematic methodology to develop a DSML?
Proposed process steps - overview

1. Capture domain knowledge
2. Define a Domain Specific Machine
   Based on a reference architecture embodying domain concepts and solution pattern
3. Linguistic Meta Model
4. Concrete syntax

Currently out of scope for our methodology
Capture knowledge – similar to use case analysis but at the level of domain
A knowledge schema to capture part of this knowledge
Capture knowledge using the schema

Eclipse Modeling Framework
Define the Domain Specific Machine

- explicate the reference architecture
A typical control systems reference architecture.
Explicate architecture into DSM in terms of domain concepts

Capture domain knowledge

1

Parent Node

Child Node

Command handling

Alarm handling

……...

Logging and archival
Authorization
Resource Management
Fault Handling
Reports

Processes

World Model

Data

Events And Alarms

Commands

Validation
Translation
Execution

Validation
Translation

Data

Events
And Alarms

Responses
Define the linguistic meta model

- apply DSL development guidelines from literature on the DSM
Guidelines from literature towards language meta model

- Need for interfaces
  - Analyze DSM, are there **entities** which require interface definition?
  - Analyze **functions, inputs, outputs** in knowledge to identify the elements of interface

- Structures from domain functions (alarm handling, command handling)
  - Can be per domain function, however the number can be large.
  - Structures to group appropriately, through modularity

- Modularity through grouping functions meaningfully
  - The entities they belong to (e.g. controller)
  - Within a group, create sub groups based on common concerns (alarms, commands, events)

- Modularity through Identifying cross cutting concerns
  - Security, Safety, Reliability
  - Each one can be a domain in its own right. Hence incorporate lite footprints
  - Else recursively apply for individual domains – perform aspect model weaving. DSL weaving is still an open area!

- Logic specification
  - Build on abstract machines
  - Reuse state machines (or similar formalisms) to capture internal behavior of structures
Experience certainty.

Language meta model - identify interface elements

Input / Output Attributes

Command

Event

State

Alarm

Input / Output Attributes as Interface Items
Domain Functions

Identifying Domain Functions and Structures

Functions

Command Handling

Command Translation

State Transition

Commands

Events

States

Inputs / Outputs
### Language meta model - grouping functions for modularity

**Command Block**
- Command Handling
- Command Translation
- State Transition

**Event Block**
- Event Generation
- Event Handling

**Grouped by common attribute**
- Command
- Event

**Domain Functions and Interface Items**
- Command Handling
- Command Translation
- State Transition
- Event Generation
- Event Handling

- Command
- Event
- State
- Event
- Action
- Alarm
Language Meta model - Identifying cross cutting concerns

- Event Block
  - Event Generation
  - Event Handling

- Command Block
  - Command Handling
  - Command Translation
  - State Transition

- Validation
  - Validate
  - is reliable event?

- Security
- Safety
- Reliability

Concerns which are out of the domain model

Extend with minimal footprint – else define independent DSLs. DSL weaving is an open area
Language meta model - logic specification

Command Block
- Command Handling
- Command Translation
- State Transition

Inline it within another block

Internal Logic for structure

E.g., State machine principles

Experience certainty.

Structures
- Process
- Interactions
- Abstract Machine

Language meta model - logic specification
DSML meta model

System
  Entity → M&C Controller
  Stakeholders → M&C Designer, Operator
  Goals → Consistent Design, Valid Operations
  Functions → Command Execution, Command Validation

M&C System
  Interface
    → Command
    → Response
    → Event
  Control Node
    → Command Block

Language meta model
Interface Description

- Scaled and used for GMRT telescope
- Used as a prototype for SKA telescope
- Adoption by the TANGO control systems community in progress
Control Systems DSL – Giant Meterwave Radio Telescope pilot

Behavior Description

```java
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 trigger conditions to
            AlarmTriggerCondition {
                DataPoints : GMRT_ID.SERVO_PWR_A
            }
            AlarmHandling {
                // Specify actions for alarms
                Action {
                    fireCommands : GMRT_ID.STOP
                    // specify the script you want to
                    Op OPI execute "File Path Of Script"
                }
            }
        }
    }
    CommandResponseBlock {
        Command GMRT_ID.HOLD {
            CommandValidation {
                parameter GMRT_ID.HOLD.paral[
                    Min Value = 0
                    Max Value = 200
                    Possible Values = (0, 200)
                ]
            }
            Transitions {
                currentState GMRT.ID.operationalManual (exitAction Action [])
                => nextState GMRT.ID.operationalAutomatic
            }
        }
        Command GMRT_ID.POSITION {
            CommandValidation {
                parameter GMRT_ID.POSITION.paral[
                    Min Value = 0
                    Max Value = 200
                    Possible Values = (0, 200)
                ]
            }
            Transitions {
                currentState GMRT_ID.Initialization
                => nextState GMRT_ID.operationalManual
            }
        }
    }
}

We also have other DSLs created using this methodology
```
Discussion and roadmap
Can it apply to other domains? Not just control systems?

- Relational database is an abstract architecture pattern
  - Spreadsheets are DSL’s to visualize data
  - SQL is to query the data
  - Both are derived from DSM and vocabulary implicitly. We are just explicating the approach.

- HTML captures the structure of documents resulting from web browsing
Future work

- Formalize the concept of DSM
- Explore tooling possibilities
- Address defining concrete syntax
- Perform empirical study of the DSL’s developed using the methodology
- Keep researching and contributing back to community (be back at DSM 😊)
  - Contributed to SysML v2

We have a granted patent on this methodology (US 10,387,124 B2)
Questions ???

For more information, contact:
Subhrojoyti Roy Chaudhuri
Subhrojoyti.c@tcs.com

Promise what we deliver.
Deliver what we promise.
That's certainty.

DSM, SPLASH 2019, Athens, Greece