Query Language for AADLv2, Requirements and first propositions

Jérôme Hugues, ISAE
Current approaches

- Analysis of AADL models rely on a QVT-like approach
  - Queries on the model, View (restriction) of model elements
  - Transform into analytic models
  - Need a language to ease implementation of analysis plug-ins
    - Reduce implementation code, hide meta-model
- REAL (O. Gilles’ PhD): based on set theory
  - One theorem = one constraint expressed on a model
    - E.g. compatibility with Ravenscar, MILS, ARINC653
- OAR (SEI, UIUC): DSL to query model elements
  - Then delegation to Mathematica to compute/refine some properties
    - E.g. allocate threads to processors
- OCL (D. Blouin): library of helper functions
Some discussions here @ Toulouse

- OCL is OK for some TopCased users
  - Checks implemented by software teams, from requirements from system engineers
- OCL is impossible to read for system engineers
  - Need a lighter syntax, close to natural language or mathematics
  - This was the design objective for REAL
- Would need also a helper library for manipulating AADLv2
  - Prototype from USB
Roadmap proposal (from Paris meeting)

- **Objectives:** do not define a DSL, but bricks of a DSL for other annex languages
- From the meta-model and OAR, OCL, REAL
  - Define mini-language for querying properties
    - Similarly, should map meta-model, and be simple
    - Reflect type and unit systems,
    - Easy to take into account new property types (this.<property_name> too complex?)
  - Define list of accessors on the **declarative** and **instance** models
    - Name of accessors should map meta-model ones,
    - But need to synchronize to reduce complexity of accessors
Objectives and non-objectives

- Main objective: *define a sublanguage to be integrated to other AADL annex languages*
- Defining a full-fledged language is out of scope
  - Use of query language depends on analysis framework, programming model, …
  - Some analysis are better expressed with ..
    - An imperative language: e.g. Response Time Analysis
    - A logic language (a-la Prolog): e.g. architectural patterns
    - A set-based language (e.g. REAL): restrictions on patterns
- A query language is the root of all these prog. Models
  - A sublanguage to fetch information from models elements
  - Should enable navigation (introspection) of models
High-Level Requirements

(R1) The query language shall have the following properties

(a) Easy integration to other language, e.g. Behavioral Annex
   ✓ E.g. compute() based on Compute_Execution_Time
(b) Follow scope rules from AADLv2
   ✓ Notion of local subcomponents, features, parent component, etc.
   ✓ Possibility to integrate in an annex language
(c) Computations made by host language

(R2) The Query Language shall support

(a) Fetching information on property values
(b) Building collections of AADL model entities
   ✓ Abstract collection: array, list, set depending on the host language
   ✓ From particular predicate (category, category + filter)
(R3) The Syntax of the Query Language shall be close to
   (a) The declarative language for manipulating property types:
       aadlinTEGER, aadlBOOLEAN, range, units, etc.
       ✓ Rationale: avoid syntactic sugar from meta-model
   (b) The meta-model for model entities: component
       type/implementation, subcomponents, features, etc.
       ✓ Rationale: take advantage of JMI code whenever possible

(R4): The syntax shall be close to Java one
   ✓ Rationale: so are JMI, OCL, ATL, et al.

• Note: divergence to R3 might be necessary to accommodate
  incidental complexities from AADL v2 meta-model
  ➢ E.g. when manipulating property types that are range of units, e.g.
    Compute_Execution_Time
Some elements of syntax
*To be adjusted when meta-model is stabilized*

- Build sets of instance model elements
  - **OAR:**
    \[
    \text{BPAllProcesses} \leftarrow \text{process:p in this};
    \text{BPAllThreads} \leftarrow \text{thread:t in BPAllProcesses};
    \]
  - **REAL:**
    \[
    \text{BPAllProcesses} = \{ \text{p in Process_Set | Is_Subcomponent (s, Root_System)} \};
    \text{BPAllThreads} = \{ \text{t in Thread_Set | Is_Subcomponent (t, BPAllProcesses)} \};
    \]

- **AADL Query Language**
  \[
  \begin{align*}
  \text{BPAllProcesses} &= \text{this.subcomponents (PROCESS)}; \\
  \text{BPAllThreads} &= \text{BPAllProcesses.subcomponents (THREAD)};
  \end{align*}
  \]
  - Dot notation + filter
  - Note: REAL supports advanced filtering, how to support this? Can we stay independent from SQL-like language here?
Some elements of syntax

To be adjusted when meta-model is stabilized

- **OAR**: typed properties
  
  ```
  p = this.Period(ms);
  c = this.Compute_Execution_Time[max](ms);
  ```

- **REAL**: unit (not yet) used
  
  ```
  var Period := Property (t, “Period”);
  ```

- **AADL Query Language**:
  
  ```
  p = this.getProperty (”Period”);
  c = last (this.getProperty (”Compute_Execution_Time”));
  ```

- Introduce operators first/last for range (from Ada), use Java-like syntax for manipulating properties that are lists, records
- Other accessors: isPropertyDefined(), propertyUnit(), ..
- Issue: managing units, including user-defined ones
Writing down requirements

• Still lot of ambiguities in the objective in the group
  ▶ PSL vs. OAR vs. LUTE vs. REAL (and BLESS, OCL …)
  ▶ Similar initiatives, need to be summarized and integrated

• Roadmap
  ▶ Carefully word requirements, the “shall” and the “won’t”
    ✓ Reduce to queries only
    ✓ To be integrated in other languages
  ▶ List requirements from typical analysis
    ✓ Covered: scheduling, security, dimensioning, matching architectural patterns, basic computations on models
    ✓ Decide when to stop, to avoid too complex queries on the model
Writing down requirements (cont’d)

• Roadmap (cont’d)
  ➢ Select a syntax: Java-like suggested, to be discussed
  ➢ Start defining accessors
    ✓ For each one, propose an illustrative example
    ✓ Help stabilizing the syntax a bit