Type System Unification

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Peter Feiler

May 2016

Type System Unification

Unification of type systems and expression languages (Peter, Lutz, Alexey, Brian, Serban)

- Alisa ReqSpec et.al.: types, assign once variables, computed variables., property types, Resolute types, Java type mapping
- Property language V3
- Constraint language
- BLESS
- Data Model annex
- Resolute, Scripting languages (Python, Ease)

Type System Unification

Types

- Data types, property types, constraint language variable types
  - Property types available as data types
  - Data types available as property types
- Base types: integer, real, string, Boolean
  - No more **aadlinteger** keyword
  - Real without .0 is accepted
- Handling of units: part of value, association via property
  - Integration of proposed Units system (ISO, SysML)
- Sequences & sets: Set with unique element semantics
- Union of types
- Type conversion: explicit casting and implicit for numerics (2 ok for real)
- Type inference from value: need for different syntax for different aggregates
- Types like time: when to limit to integer vs. allowing real

Type System Unification Approach

Common type system available for use as data types, property types, annex sublanguage types

- Types can have properties

Base types

- Numeric, Boolean, string, enumeration, units

User defined types

- ~~**Data** int16 **extends** Integer { Data_Size => 16 Bits};~~
- Int16: **type** Integer { Data_Size => 16 Bits};
- Temperature: **type** real **units** Celsius;
- Speed: **type** integer [0 .. 200 kph] **units** SpeedUnits;

Composite types

- Unions and aggregates
- Aggregates: records, arrays, sets, list, sequences
- Personel_Record: **record** (first: string; last: Address;);

Type System Unification Approach

Data component types

User defined types

- Data Personnel_Record
- Features

Multiple implementations for type? No

Type System Usage

Port types

- P1: **in data port** Temperature;

Data components

- DataObject: **data** Personel_Record;
- Data type could have multiple implementations
  - Substitution of subtypes vs. arbitrary implementations

Properties

- Property definitions reference types

Data Annex

- Characterization via properties vs. partial specification
- **Data** personel_record { Data_Representation => Struct; };
- Personel_Record: **type record** () { Source_Name => PersonnelRecord;};

- Personel_Record: **refined to type record** (first: string; last : string;);

Meta Data: Modeling Related Properties

Relevant for structural constraint languages

Reference types

- References to model elements

Meta model classes

- Currently we have classifier

Expressions

- Currently not part of property sublanguage
- Operators
- Built-in functions
- User definable functions

- Behavioral & temporal specifications