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
- Handling of units: part of value, association via property
  - Integration of proposed Units system (ISO, SysML)
  - Unit assumptions vs. units passed as part of value
- Sequences & sets: Set with unique element semantics
- Union of types
- Type conversion: explicit casting and implicit for numerics
  - Real without .0 is accepted
  - Numeric and numeric range
- Types like time: when to use integer vs. real
- Support for type inference from value? Require type

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

- 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, multiset (bag), list(sequences), map, graph
- Personel_Record: **record** (first: string; last: Address;);

Provide subtyping?

Type System Unification Approach

Data component types

- Component type syntax vs. type system syntax?
- Multiple implementations for type? No

Arrays of components and features in core language. Relation to arrays in type system.

Type System Usage

Type matching rules exist for classifiers.

Issue of visibility rules for types vs. those for data classifiers.

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