

# Type System Unification

Software Engineering Institute  
Carnegie Mellon University  
Pittsburgh, PA 15213

Peter Feiler

Jan 2017



# 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

- Same as subclassing
- Equivalent of type extension for data types



# Type System Usage

## Port types

- P1: **in port** Temperature;

**Type matching rules exist for classifiers.**

## Data components

- DataObject: **data** Personel\_Record;
  - Subtype substitution/match (Type\_Extension)

## 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;);



# Type System Unification Approach

## Data component implementations

- Blueprint is aggregate structure
- Multiple implementations for type? No
  - Subtypes & union types

Denis example of protocol wrappers can be addressed by union type.

**TBC: Arrays of components and features in core language. Harmonization with arrays in type system.**



# System Properties: Property Sets & Stereotypes

An attribute of the system or system element

Type definitions in packages or in property sets

- Packages provide dot separated IDs as names
  - Request to offer the same for property sets
- Property sets have simple ID as name
  - Grouping of properties for common purpose under simple name

Property stereotype

- Specification of which property definitions apply to a component
  - May or may not include a property value
  - Gets associated with component classifier (or other model element)
- Component can have multiple associated stereo types
  - Property definition reference in multiple stereo types is acceptable
  - Specification of which properties are relevant for given analysis



# Meta Data: Model/Design Related Properties

Structural design constraints

V2 properties in Modeling property set

- 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 design constraints

- Should we use PSL?

