Using Physical Quantities in Robot Software Models

Loli Burgueño\textsuperscript{a}, Tanja Mayerhofer\textsuperscript{b}, Manuel Wimmer\textsuperscript{b}, Antonio Vallecillo\textsuperscript{a}

\textsuperscript{a}Atenea Research Group, Univ. Málaga, Spain
\textsuperscript{b}Business Informatics Group, TU Wien, Austria

RoSE’18, May 28, 2018, Gothenburg, Sweden
Models

Abstraction vs Precision

“Being abstract is something profoundly different from being vague... The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise.”

Edsger Dijkstra
The abstract-o-meter
by Christoph Niemann (http://www.christophniemann.com)

Challenge in software development: deciding the right level of abstraction for every task

https://modeling-languages.com/essential-software-engineering-quotes-on-instagram
Motivation

Uncertainty and Units in Engineering Disciplines

- Engineers naturally think about:
  - *uncertainty* associated with *measured values* and,
  - *units of values"

- Uncertainty and units are explicitly defined in their models and considered in model-based simulations

Example: Coupled Clutches of Modelica Standard Library

(Coupled Clutches Example of Modelica Standard Library)
However the situation is not the same when modeled in software! 😞
Measurement Uncertainty in Software Models – Some Attempts

code:

```
context RectangleSw::area() : Real = h*w
```

```plantuml
class RectangleSw {
  +x : Real
  +y : Real
  +h : Real
  +w : Real
  +area() : Real
}

class RectangleEng {
  +x_Min : Real
  +x_Max : Real
  +y_Min : Real
  +y_Max : Real
  +h_Min : Real
  +h_Max : Real
  +w_Min : Real
  +w_Max : Real
  +area_Max() : Real
  +area_Min() : Real
}

class RectangleSw1 {
  «uncertain»+x : Real
  «uncertain»+y : Real
  «uncertain»+h : Real
  «uncertain»+w : Real
  «uncertain»+area() : Real
}
```

```
+ stereotype uncertain
+ precision : Real
```
Motivation

Uncertainty and Units in Software Engineering

- Very limited support for representing uncertainty and units in software models
- No support for considering such properties in model-based simulations
- Not part of their type systems!

<table>
<thead>
<tr>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>value : Real</td>
</tr>
</tbody>
</table>

What kind of value is measured?

In which unit is the value measured?

What is the uncertainty of the measurement method?
Example: The way we would like (and we do) model

\[
\text{duration} = \text{end.time} - \text{start.time} \\
\text{distance} = \text{end.position} - \text{start.position} \\
\text{avgVelocity} = \frac{\text{distance}}{\text{duration}} \\
\text{avgAcceleration} = \frac{\text{end.velocity} - \text{start.velocity}}{\text{duration}}
\]
Example: The way we would like to model with units and uncertain data

**S1 : Segment**
- /duration = 10.0 ± 0.0019799 s
- /distance = 10.0 ± 0.0014142 m
- /avgVelocity = 1.0000000392 ± 0.000489 m/s
- /avgAcceleration = 0.200000008 ± 0.0632468 m/s²

**M0 : Observation**
- time = 0.0 ± 0.0014 s
- position = 0.0 ± 0.001 m
- velocity = 0.0 m/s

**M1 : Observation**
- time = 10.0 ± 0.0014 s
- position = 10.0 ± 0.001 m
- velocity = 2.0 ± 0.02 m/s
A Quantity is an observable property of an object, event or system that can be measured and quantified numerically. [QUDT]
- For example its length, mass, speed or temperature

[NIST SI] Quantities are composed by:
- a **magnitude** expressed as a number—usually accompanied by an uncertainty
- a **unit**, which establishes the reference point in which the quantity is described
  - For example, 3.5 m/s
Units (and Dimensions)

International System of Units (SI, ISO 80000)

- **Base dimensions**: Length, Mass, Time, Electric Current, Thermodynamic Temperature, Amount of Substance, Luminous Intensity, Data Storage Capacity, Entropy, Traffic Intensity, Level, Angle
- **Base units**: Meter (m), Kilogram (kg), Second (s), Ampere (A), Kelvin (K), Mole (mol), Candela (cd), Bit (b), Shannon (Sh), Erlang (E), Decibel (dB), Radian (rad)
- **Derived dimensions**: 100+ dimensions
  - e.g., Area, Volume, Linear Velocity.
- **Derived units**: 100+ units
  - e.g., Square Meter (m²), Cubic Meter (m³), Meter per Second (m/s)

Other Systems of Units

- Centimeter-Gram-Second System (CGS)
- Imperial System
- United States Customary System (USCS, USC)

Quantities

- Mass
- Time
- Electric Current
- Thermodynamic Temperature
- Amount Of Substance
- Luminosity Intensity
- Angle
- Linear Velocity
- Linear Acceleration
- Power
- Resistance
- Force

Unit
- name : String
- symbol : String
- ...
Quantity types for base dimensions

Length
Mass
Time

Quantity types for derived dimensions

AmountOfSubstance
Luminosity
Intensity
Angle

LinearVelocity

DimensionlessQuantity

Power

LinearAcceleration

Force

Resistance

电势

电压

电流

温度

物质的量

光度

强度

角度

线性加速度

力

电功率

长度

加(l: Length) : Length

减(l: Length) : Length

乘(l: Length) : Area

乘(l: Area) : Volume

除以(t: Time) : LinearVelocity

...
Units

Model-Based Representation

- **Unit**
  - name : String
  - symbol : String
  - dimensions : Real [12]
  - conversionFactor : Real [12]
  - offset : Real [12]
  - isBaseUnit() : Boolean
  - isCoherentDerivedUnit() : Boolean
  - isDimensionlessUnit() : Boolean
  - isCompatibleWith(Unit u) : Boolean
  - equals(Unit u) : Boolean
  - multiplyUnits(Unit u) : Unit
  - divideUnits(Unit u) : Unit
  - powerUnits(Real s) : Unit

**Query nature of unit**

- isBaseUnit()
- isCoherentDerivedUnit()
- isDimensionlessUnit()

**Compare units**

- isCompatibleWith(Unit u)

**Combine units**

- multiplyUnits(Unit u)
- divideUnits(Unit u)
- powerUnits(Real s)

0..* multiple
0..* submultiple
Definition: Standard Uncertainty [GUM]

- Uncertainty of the result of a measurement $x$ expressed as a standard deviation $u$
- Representation: $x \pm u$ or $(x, u)$
- Examples:
  Normal distribution: $(x, \sigma)$ with mean $x$, standard deviation $\sigma$
  Interval $[a, b]$: Uniform or rectangular distribution is assumed
    
    $$(x, u) \text{ with } x = \frac{a+b}{2}, \ u = \frac{(b-a)}{2\sqrt{3}}$$

Computations with uncertain values have to respect the *propagation of uncertainty* (uncertainty analysis)

Two Methods for Computing Aggregated Uncertainty

- Normal or Uniform distribution: Analytical (closed-form) solutions
- General case: Using samples (SIPMath Std.)

Our Work

1. **Type system** for representing measurement uncertainty and units
   - Kernel representation for Quantities (value + uncertainty)
   - All SI *Dimensions* and their operations available
   - All SI Units available

2. **Algebra of operations** for performing computations with units and uncertain data
   - Computational kernel for computing quantities
   - Type-safe operations
     - Length + Length returns a Length
     - Length / Time returns a LinearVelocity
     - Length + Time is not valid
     - *Independently from whether the Length is expressed in meters, feet or inches*
   - Propagation of uncertainty

3. **Implementations** in UML, OCL, fUML y Java
   - Download: [https://github.com/moliz/moliz.quantitytypes](https://github.com/moliz/moliz.quantitytypes)
Quantities in UML Models (MARTE and SysML)

- MARTE has stereotypes to decorate values with information about the units they are expressed in, and with measurement uncertainty (“precision”)
  - However:
    - It is simply “decorative information”: no type checking, no operations for aggregating uncertainty values
    - Units used as types! (not “Dimensions as types”)

- SysML 1.4 provides the QUDV (Quantities, Units, Dimensions) and ISO 80000 library with all units and dimensions.
  - However:
    - No support for dealing with measurement uncertainty
    - Types of values are still based on explicit units

Compatibilty problems when combining MARTE and SysML models

NOTE: Of course, simulation tools (Modelica, Matlab/Simulink) and mathematic languages (Mathematica) provide support for units, dimensions and uncertainty, but they are at a different abstraction level
Ozobot & Ozoblockly

1. The robot is able to perform two movements:
   a. **Rotate** its head
   b. **Move** in the direction its head points to

2. A **plan** is composed by a sequence of movements

3. The robot can be assigned a **mission**—which determines the target position it is supposed to reach with a plan
Modeling the robot with USE

Robot
position : Coordinate
headsTo : Real
performAllMoves()
Analysis

1. Simulation

2. Verification
Behavior in USE

class Movement
attributes
move:Length
rotate:Angle
operations
performMove()
begin
   declare aux:Coordinate,sa:UReal,ca:UReal,dx:Length,dy:Length;
   -- we change the angle first (if we have to)
   if not self.rotate.oclIsUndefined() then
      self.robot.headsTo := self.rotate
   end;
   -- and then we move (if we have to)
   if not self.move.oclIsUndefined() then
      ca := self.robot.headsTo.cos();
      sa := self.robot.headsTo.sin();
      dx := self.move * ca;
      dy := self.move * sa;
      aux := new Coordinate;
      aux.x := self.robot.position.x + dx;
      aux.y := self.robot.position.y + dy;
      self.robot.position := aux;
   end;
end
end
Executing the USE specs

Robot.soil> !new Coordinate('initial')
Robot.soil> !initial.x:=x00
Robot.soil> !initial.y:=y00
Robot.soil> !new Coordinate('target')
Robot.soil> !target.x:=x10
Robot.soil> !target.y:=y10
Robot.soil> !new Robot('robot')
Robot.soil> !robot.position:=initial
Robot.soil> !robot.headsTo:=h0
Robot.soil> !new Movement('m1')
Robot.soil> !m1.move:=x10
Robot.soil> !new Movement('m2')
Robot.soil> !m2.rotate:=h90
Robot.soil> !new Movement('m3')
Robot.soil> !m3.move:=x10
Robot.soil> !new Movement('m4')
Robot.soil> !m4.rotate:=h225
Robot.soil> !m4.move:=xy1010
Robot.soil> !new Movement('m5')
Robot.soil> !m5.rotate:=h45
Robot.soil> !m5.move:=xy1010
Robot.soil> !insert(robot,m1) into Plan
Robot.soil> !insert(robot,m2) into Plan
Robot.soil> !insert(robot,m3) into Plan
Robot.soil> !insert(robot,m4) into Plan
Robot.soil> !insert(robot,m5) into Plan
Robot.soil> !robot.performAllMoves()
Robot.soil> !r:=robot.position.uCoincide(target)
Robot.soil> ?r

→ 0.03479626661931806 : Real
Object diagram with the resulting system
Ongoing and Future Work

- **Implementation**
  - Evolve fUML proof-of-concept implementation to full implementation
  - Alf implementation (textual action language for fUML)
  - Full integration with Papyrus (already done for MagicDraw)
  - Integration with the OCL/USE type system **Done**

- **Refinement of the conceptual model of quantity types**
  - Different kinds of uncertainty (e.g., interval, different probability distributions)

- **Representation of quantities**
  - Integration with existing standards, e.g., MARTE and SysML
  - Connection with other analysis/simulation tools (Modelica, Simulink, etc.)

- **More extensive case studies**
  - Larger case studies to evaluate the usability and effectiveness of our notation
  - Industrial case studies to analyze the propagation of uncertainty in real situations
Using Physical Quantities in Robot Software Models

Loli Burgueño\textsuperscript{a}, Tanja Mayerhofer\textsuperscript{b}, Manuel Wimmer\textsuperscript{b}, Antonio Vallecillo\textsuperscript{a}

\textsuperscript{a}Atenea Research Group, Univ. Málaga, Spain
\textsuperscript{b}Business Informatics Group, TU Wien, Austria

RoSE’18, May 28, 2018, Gothenburg, Sweden