Uncertainty Modeling Framework for the Integration Level V.4

All the sections excluding Section 3.4: Man Zhang, Shaukat Ali, Tao Yue
manzhang@simula.no, shaukat@simula.no, tao@simula.no
Section 3.4: Phu Hong Nguyen
phu@simula.no
Simula Research Laboratory, Norway
Contact Person: Man Zhang (manzhang@simula.no)

Executive Summary

This document describes the Uncertainty Modeling Framework for modeling integration level uncertainties in the context of Cyber-Physical Systems (CPS). Section 1 presents an overview of the framework, Section 2 documents a UML profile for modeling uncertainties, referred to as UML Uncertainty Profile (UUP), Section 3 presents various model libraries related to uncertainty modeling, and Section 4 presents our modeling methodology.

1 OVERVIEW OF CPS UNCERTAINTY MODELING FRAMEWORK

An overview of our CPS *Uncertainty Modeling Framework* (UncerTum) is shown in Figure 1. Notice that our framework is exclusively developed to support developing test ready models together with uncertainty to facilitate model-based testing. This means that our framework doesn't support modeling of CPS and uncertainty from the design and development perspective rather just supports modeling only at the level of detail that is required to facilitate model-based testing.

At the core of the framework is the implementation of *U-Model* [1]. More specifically, the core part of the *U-Model* is implemented as a UML Profile comprising of three UML profiles including *Belief* profile, *Uncertainty* profile, *IndeterminacySource* profile and *Measurement* profile. All these profiles import *Internal UUP* library that defines the necessary Enumerations required in the profiles. The framework also consists of a *CPS* UML profile that permits modeling the specific aspects of the three levels of CPS, i.e., *Application*, *Infrastructure*, and *Integration* required for supporting testing at the three levels.

The framework also consists of four UML model libraries including *Risk Assessment Library*, *Measure Library, Pattern Library*, and *Time Library* that extend the UML MARTE profile [2]. The framework relies on the UML Testing Profile V.2 to bring the MBT concepts into the models to support testing. Finally, the framework provides a set of guidelines to use the overall framework to model uncertainty in CPS at the three levels to support testing.

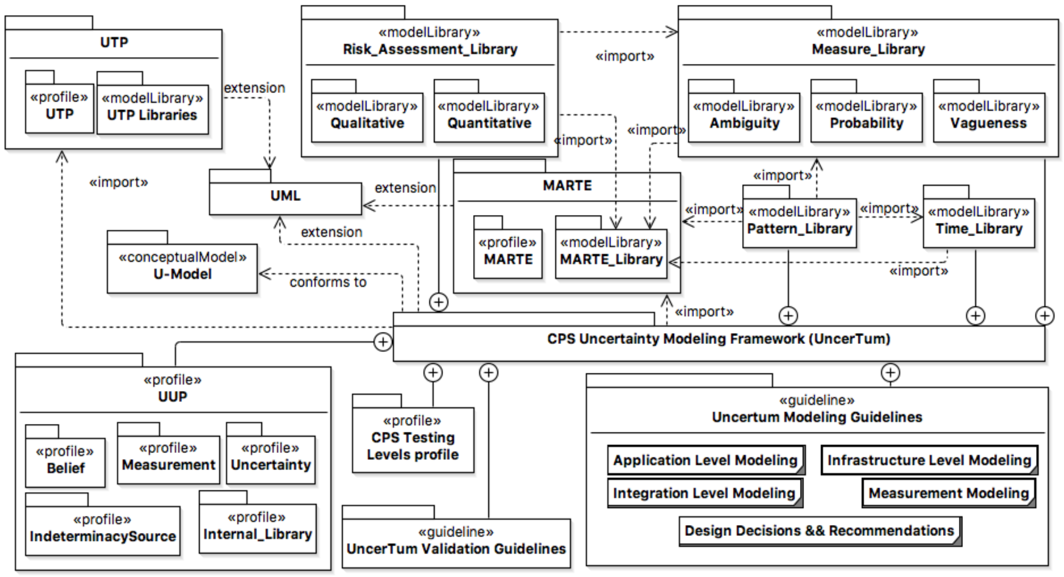
UTP
«profile» UTP
«modelLibrary» UTP Libraries
extension
«modelLibrary» Risk_Assessment_Library
«modelLibrary» Qualitative
«modelLibrary» Quantitative
«import»
«modelLibrary» Measure_Library
«modelLibrary» Ambiguity
«modelLibrary» Probability
«modelLibrary» Vagueness
«import»
UML
extension
MARTE
«profile» MARTE
«modelLibrary» MARTE_Library
«modelLibrary» Pattern_Library
«modelLibrary» Time_Library
extension
«conceptualModel» U-Model
conforms to
CPS Uncertainty Modeling Framework (UncerTum)
«profile» UUP
«profile» Belief
«profile» Measurement
«profile» Uncertainty
«profile» IndeterminacySource
«profile» Internal_Library
«profile» CPS Testing Levels profile
«guideline» UncerTum Validation Guidelines
«guideline» Uncertum Modeling Guidelines
Application Level Modeling
Infrastructure Level Modeling
Integration Level Modeling
Measurement Modeling
Design Decisions && Recommendations

Figure 1. Overview of CPS Uncertainty Modeling Framework

2 Profile Specification

This section describes the core uncertainty profile of the *Uncertainty Modeling Framework* (*UncerTum*).

2.1 UML Uncertainty Profile (UUP)

This section presents the stereotypes and classes that are part of the *UUP* and are shown in Figure 2.

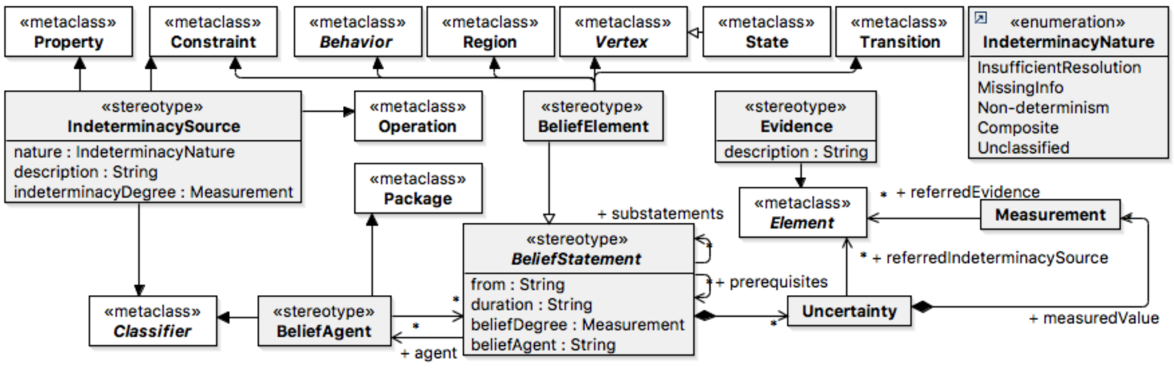

Figure 2. UML profile diagram for Belief modeling

2.1.1 Belief Profile

2.1.1.1 «BeliefStatement»

This abstract stereotype implements the semantics of the U-Model::BeliefModel::BeliefStatement concept [1]. As defined in [1] *"A Belief is an implicit subjective explanation or description of some phenomena or notions held by a BeliefAgent"*. The *Belief* is an abstract concept that is usually held by one or more belief agents and such belief is concretized as one or more belief statements. However, in terms of modeling, a model consists of a set of model elements and this is the reason that we keep «BeliefStatement» as abstract and specialized it into «BeliefElement», which is discussed next.

Extensions

- None

Generalizations

- None

Associations

- agent: «BeliefAgent» [*]

  Optional many agents who hold a common belief statement
- substatements: «BeliefStatement» [*]

  Optionally a belief statement refers to a set of sub-statements
- prerequisites: «BeliefStatement» [*]

  The set of belief statements, on which this belief statement depends
- uncertainty: Uncertainty [*]

  Optional many uncertainties associated with the belief statement

Attributes

- from: String[0..1]

  Optional attribute to specify the time at which belief agent(s) made the belief statement
- duration: String[0..1]

  Optional attribute to specify the duration of time for which the belief statement remains valid
- beliefDegree: Measurement [*]

  Optional many measurements to specify the degree of belief the belief agents hold about the belief statement. More details can be consulted in [1].
- beliefAgent: String[*]

  Optional specification of a set of belief agents as a set of strings, who made this belief statement.

Constraints

[1] *Belief Agent Constraint:* There is at least one belief agent for a belief statement.

```
self.agent->size() +self.beliefAgent->size() >=1
```

[2] *Belief Degree Measure Constraint:* Each belief degree associated with the belief statement must have an associated measure.

```
self.beliefDegree->size() >0 and self.beliefDegree->select(measurement:Measurement|
measurement->size()>0)->forAll(measurement:Measurement|(measurement.measureInDT->size()+
measurement.measureInDTViaClass->size()=1) xor (not measurement.measure.oclIsUndefined()))
```

[3] *Belief Degree Measurement Constraint*: Each belief degree associated with the belief statement must have an associated measurement.

```
self.beliefDegree->size() >0 and self.beliefDegree->select(measurement:Measurement|
measurement->size()>0)->forAll(measurement:Measurement| not (measurement.measurementInVS.
oclIsUndefined() xor measurement.measurement.oclIsUndefined() xor measurement.
measurement.oclIsUndefined()))
```

[4] *Belief Degree Measurement and Measure Constraint*: Each belief degree measurement associated with the belief statement must have a measure and associated measurement.

```
self.beliefDegree->size() >0 and self.beliefDegree->select(measurement:Measurement|
measurement->size()>0)->forAll(measurement:Measurement|(measurement.measureInDT-
>size()+measurement.measureInDTViaClass->size() + measurement.measure->size()) =
(measurement.measurement->size()+measurement.measurementInVS->size()))
```

[5] *Uncertainty Measure Constraint:* Each uncertainty associated with the belief statement must have an associated measure.

```
self.uncertainty->size() >0 and self.uncertainty->select(uncertainty:Uncertainty|
uncertainty.uncertaintyMeasurement->size()>0)->forAll(uncertainty:Uncertainty|uncertainty.
uncertaintyMeasurement->forAll(measurement:Measurement| not measurement.measurementInVS.
oclIsUndefined() xor not measurement.measurement.oclIsUndefined()))
```

[6] *Uncertainty Measurement Constraint:* Each uncertainty associated with the belief statement must have an associated measurement.

```
self.uncertainty->size() >0 and self.uncertainty->select(uncertainty:Uncertainty|
uncertainty.uncertaintyMeasurement->size()>0)->forAll(uncertainty:Uncertainty|uncertainty.
uncertaintyMeasurement->forAll(measurement:Measurement|measurement.measureInDT->size()+
measurement.measureInDTViaClass->size() =1 xor not measurement.measure.oclIsUndefined()))
```

[7] *Uncertainty Measurement and Measure Constraint*: Each uncertainty measurement associated with the belief statement must have a measure and its associated measurement.

```
self.uncertainty->size() >0 and self.uncertainty->select(uncertainty:Uncertainty|
uncertainty.uncertaintyMeasurement->size()>0)->forAll(uncertainty:Uncertainty|
uncertainty.uncertaintyMeasurement->forAll(measurement:Measurement|(measurement.measureInDT-
>size()+measurement.measureInDTViaClass->size() + measurement.measure->size())=
(measurement.measurement->size()+measurement.measurementInVS->size())))
```

2.1.1.2 «BeliefElement»

This stereotype is specialization of «BeliefStatement» that is more relevant in the context of UML and modeling in general.

Extensions

- uml::Behavior
- uml::Constraint
- uml::Region
- uml::Vertex
- uml::Transition

Generalizations

- «BeliefStatement»

Associations

- None

Attributes

- None

Constraints

- None

2.1.1.3 «BeliefAgent»

This stereotype implements the U-Model::BeliefModel::BeliefAgent concept as defined in [1]. As defined in [1], "A BeliefAgent is a physical entity owning one or more Beliefs about phenomena/notion".

Different methodologies might be used to specify a BeliefAgent. For example, if a BeliefAgent is specified as a Class, then it is possible to describe its behavior (if needed) using a state machine for example. If a BeliefAgent is captured as a Package, then it is just a categorization. In the methodology section (Section 4), we present the possible ways belief agents can be modeled.

Extensions

- uml::Classifier
- uml::Package

Generalizations

- None

Associations

- beliefStatement: «BeliefStatement» [*]

  Optional many belief statements held by a belief agent

Attributes

- None

Constraints

- None

2.1.1.4 «IndeterminacySource»

This stereotype implements the U-Model::BeliefModel::IndeterminacySource concept [1]. As defined in [1], "It represents a situation whereby the information required ascertaining the validity of a BeliefStatement is indeterminate in some way, resulting in Uncertainty being associated with that statement."

Extensions

- uml::Classifier
- uml::Constraint
- uml::Property
- uml::Operation

Generalizations

- None

Associations

- None

Attributes

- description: String[0..1]

  An optional description of the indeterminacy source as string
- nature: «IndeterminacySource» [1]

  Nature of the indeterminacy source as defined in the U-Model and further elaborated in Section 2.1.1.4.
- indeterminacyDegree: Measurement[*]

  Optional many measurements associated with the indeterminacy source
- specs: Constraint[*]

  Optional many constraints that describes the possible occurrences associated with the indeterminacy source

Constraints

[1] *Indeterminacy Source Measure Constraint:* Each measure associated with the indeterminacy source must have an associated measure.

```
self.indeterminacyDegree->size()    >0    and    self.indeterminacyDegree->select(measurement:
Measurement|  measurement->size()>0)->forAll(  measurement:Measurement|measurement.measureInDT-
>size()+measurement.measureInDTViaClass->size()     =1     xor     not     measurement.measure
.oclIsUndefined())
```

[2] *Indeterminacy Source Measurement Constraint:* Each measurements associated with the indeterminacy source must have corresponding measurements specified.

```
self.indeterminacyDegree->size()    >0    and    self.indeterminacyDegree->select(measurement:
Measurement|     measurement->size()>0)->forAll(measurement:Measurement|     not     measurement.
measurementInVS.oclIsUndefined() xor not measurement.measurement.oclIsUndefined())
```

[3] *Indeterminacy Source Measurement and Measure Constraint:* Each measurement associated with the indeterminacy source must have a measure and its associated measurement.

```
self.indeterminacyDegree->size()    >0    and    self.indeterminacyDegree->select(measurement
:Measurement|  measurement->size()>0)->forAll(measurement:Measurement  |(measurement.measureInDT
->size()+measurement.measureInDTViaClass->size()+measurement.measure->size())=(measurement.
measurement->size()+measurement.measurementInVS->size()))
```

2.1.2 Uncertainty Profile

This section presents the Uncertainty Profile and the profile diagram is shown in Figure 3 followed by the detailed descriptions of stereotypes and classes.

2.1.2.1 «Cause»

Anything from which an Uncertainty occurs in the BeliefStatement. The cause for an Uncertainty can be: 1) another known Uncertainty, 2) something known and is not Uncertainty, 3) anything unknown. If a Cause is Uncertainty, then it may be measured using Measurement.

Extensions

- uml::Element

Generalizations

- None

Associations

- None

Attributes

- description: String[0..1]

  An optional description of the cause of an uncertainty as a string.

Constraints

- None

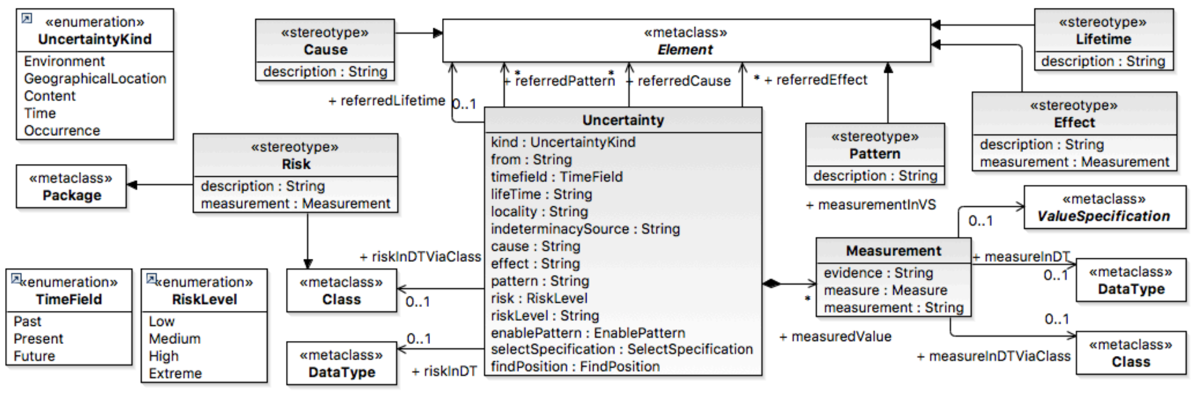

Figure 3. UML profile diagram for Uncertainty modeling

2.1.2.2 «Effect»

This is the concept from U-Model::UncertaintyModel::Effect [1]. Effect represents the result of Uncertainty in the BeliefStatement. An uncertainty may result into: 1) another known Uncertainty, 2) something known and is not Uncertainty, 3) anything unknown.

Extensions

- uml::Element

Generalizations

- None

Associations

- None

Attributes

- description: String[0..1]

  An optional description of the effect as a string.
- measurement: Measurement[*]

  Optional many measurements associated with the effect

Constraints

[1] *Effect Measure Constraint:* Each measure associated with the effect must have a corresponding measure.

```
self.measurement->size() >0 and self.measurement->select (measurement:Measurement|
measurement->size()>0)->forAll(measurement:Measurement|measurement.measureInDT->size()+
measurement.measureInDTViaClass->size() =1 xor not measurement.measure.oclIsUndefined())
```

[2] *Effect Measurement Constraint:* Each measurement associated with the effect must have corresponding measurement specified.

```
self.measurement->size() >0 and self.measurement->select(measurement:Measurement| measurement-
>size()>0)->forAll(measurement:Measurement| not measurement.measurementInVS. oclIsUndefined()
xor not measurement.measurement.oclIsUndefined())
```

[3] *Effect Measurement and Measure Constraint:* Each measurement associated with the effect must have associated measure and associated measurement.

```
self.measurement->size() >0 and self.measurement->select(measurement:Measurement| measurement-
>size()>0)->forAll(measurement:Measurement|(measurement.measureInDT-
>size()+measurement.measureInDTViaClass->size() =1 + measurement.measure->size()) =
(measurement.measurement->size()+measurement.measurementInVS->size()))
```

2.1.2.3 «Evidence»

This stereotype implements the semantics of the U-Model::BeliefModel::Evidence concept [1]. As defined in [1], "Evidence is either an observation or a record of a real-world event occurrence or, alternatively, the conclusion of some formalized chain of logical inference that provides information that can contribute to determining the validity (i.e., truthfulness) of a BeliefStatement."

Extensions

- uml::Element

Generalizations

- None

Associations

- None

Attributes

- description: String[0..1]

  An optional description of the evidence as a string

Constraints

- None

2.1.2.4 «Lifetime»

This stereotype implements the U-Model::UncertaintyModel::Lifetime concept [1]. As defined in [1], "Lifetime represents an interval of time, during which an Uncertainty exists".

Extensions

- uml::Element

Generalizations

- None

Associations

- None

Attributes

- description: String[0..1]

  An optional description of the lifetime as a string

Constraints

- None

2.1.2.5 Measurement

This is the concept adopted from the U-Model::BeliefModel::Measurement concept [1]. As defined in [1], "Measurement expressing in some concrete form the *subjective degree* of uncertainty held by the agent to a BeliefStatement".

Extensions

- None

Generalizations

- None

Associations

- measureInDTViaClass: uml::Class[0..1]

  An optional specification of measurement as a UML Class
- measurementInDT: uml::DataType[0..1]

  An optional specification of measurement as a UML DataType
- measurementInVS: uml::ValueSpecification[0..1]

  An optional specification of measurement as a UML ValueSpecification
- referredEvidence: uml::Element[*]

  An optional manual specification of Evidence as any UML Element

Attributes

- evidence: String[*]

  Optional many specification of Evidence as a set of String
- measure: Measure[0..1]

  Optional specification of measure associated with the Measurement
- measurement: String[0..1]

  An optional specification of the Measurement as a String

Constraints

- None

2.1.2.6 «Pattern»

This stereotype implements the semantics of the U-Model::UncertaintyModel::Pattern concept [1]. A pattern represents a particular pattern in which an uncertainty can occur.

Extensions

- uml::Element

Generalizations

- None

Associations

- None

Attributes

- description: String[0..1]

  An optional description of the pattern as a string

Constraints

- None

2.1.2.7 «Risk»

This concept implements the U-Model::UncertaintyModel::Risk concept [1].

Extensions

- uml::Package
- uml::Class

Generalizations

- None

Associations

- None

Attributes

- description: String[0..1]

  An optional description of the risk as a string.
- measurement: Measurement[*]

  Optional many measurements associated with the risk

Constraints

[1] *Risk Measure Constraint:* Each measure associated with the risk must have a measure specified.

```
self.measurement->size() >0 and self.measurement->select(measurement:Measurement| measurement-
>size()>0)->forAll(measurement:Measurement|measurement.measureInDT->size()+
measurement.measureInDTViaClass->size() =1 xor not measurement.measure.oclIsUndefined())
```

[2] *Risk Measurement Constraint:* Each measurement associated with the risk must have an associated measurement.

```
self.measurement->size() >0 and self.measurement->select(measurement:Measurement| measurement-
>size()>0)->forAll(measurement:Measurement| not measurement.measurementInVS. oclIsUndefined()
xor not measurement.measurement.oclIsUndefined())
```

[3] *Risk Measurement and Measure Constraint:* Each measurement associated with Risk must have an associated measure and measurement.

```
self.measurement->size() >0 and self.measurement->select(measurement:Measurement| measurement-
>size()>0)->forAll(measurement:Measurement|(measurement.measureInDT-
>size()+measurement.measureInDTViaClass->size()+measurement.measure->size())=
(measurement.measurement->size()+measurement.measurementInVS->size()))
```

2.1.2.8 Uncertainty

This is the concept adopted from the U-Model::BeliefModel::Uncertainty concept [1]. As defined in [1], "uncertainty is a *state* (i.e., worldview) of some agent or agency – henceforth referred to as a BeliefAgent – that, for whatever reason, is incapable of possessing complete and fully accurate knowledge about some subject of interest."

Extensions

- None

Generalizations

- None

Associations

- riskInDTViaClass: uml::Class[0..1]

  An optional specification of risk as a UML Class associated with the uncertainty
- riskInDT: uml::DataType[0..1]

  An optional specification of risk as a UML DataType associated with the uncertainty
- referredLifetime: uml::Element[0..1]

  An optional specification of Lifetime as a UML Element associated with the uncertainty
- referredCause: uml::Element[0..1]

  An optional specification of Cause as a UML Element associated with the uncertainty
- referredEffect: uml::Element[0..1]

  An optional specification of Effect as a UML Element associated with the uncertainty
- referredPattern: uml::Element[0..1]

  An optional specification of Pattern as a UML Element associated with the uncertainty
- referredIndeterminacySource: Element[*]

An optional many specifications of IndeterminacySource associated with the uncertainty
- uncertaintyMeasurement: Measurement[*]
  Optional many specifications of Measurement associated with the uncertainty

Attributes
- kind: UncertaintyKind[1]
  Specification of kind of uncertainty as one of the literals in the UncertaintyKind enumeration (Section 2.2.9)
- from: String[0..1]
  Optional attribute to specify the point in time at which the uncertainty specified
- field: TimeField[1]
  This value is used for identifying a relative point in time when this Uncertainty exists
- lifeTime: String[0..1]
  An optional specification of the Lifetime of uncertainty as a String
- locality: String[0..1]
  An optional specification of the Locality of the uncertainty as a String
- indeterminacySource: String[0..1]
  An optional specification of IndeterminacySource as a String
- cause: String[0..1]
  An optional specification of Cause as a String
- effect: String[0..1]
  An optional specification of Effect as a String
- pattern: String[0..1]
  An optional specification of Pattern as a String
- risk: RiskLevel[0..1]
  An optional specification of the risk level associated with the uncertainty as one of the literals from the RiskLevel enumeration (Section 2.2.7)
- riskLevel: String[0..1]
  An optional specification of the risk level associated with the uncertainty as a String
- findPosition: FindPosition [0..1]
  The position to enable this indeterminacy source.
- enablePattern: EnablePattern[0..1]
  The pattern to enable the associated indeterminacy source.
- selectSpecification: EnableSpecification[0..1]
  The specification to enable the associated indeterminacy source, e.g. *All* – all associated indeterminacy specifications are enabled.

Constraints
- None

2.1.3 IndeterminacySource Profile

This section presents the UML profile for the indeterminacy source. The corresponding profile diagram is shown in Figure 4.

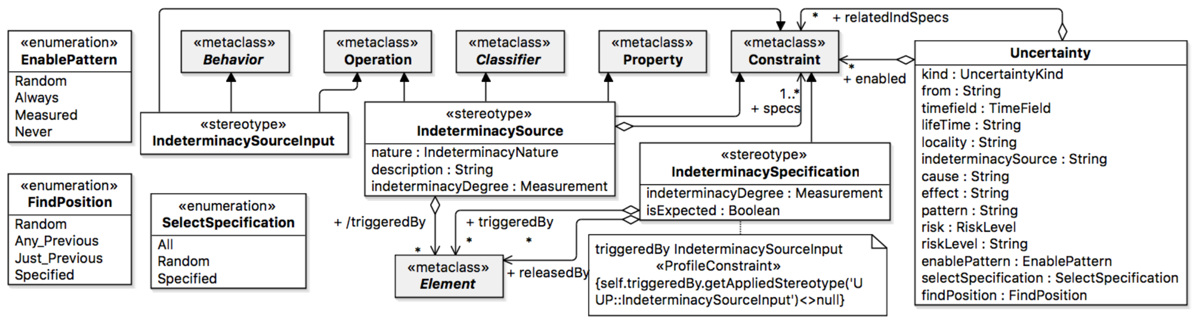

Figure 4. UML profile diagram for IndetermiancySource modeling

2.1.3.1 IndeterminacySourceInput

This stereotype is implemented to specify the action that triggers the occurrence of «IndeterminacySource».

Extensions

- uml::Operation
- uml::Behavior

Generalizations

- None

Associations

- None

Attributes

- None

Constraints

- None

2.1.3.2 IndeterminacySourceSpecification

This stereotype is implemented to specify the condition that must be true for an indeterminacy source to occur.

Extensions

- uml::Constraint

Generalizations

- None

Associations

- triggeredBy: Element [*]

  Optional action that triggers the occurrence of associated indeterminacy specification.
- releasedBy: Element [*]

  Optional action that releases the occurrence of associated indeterminacy specification.

Attributes

- indeterminacyDegree: Measurement[0..1]

  Optional specifications of Measurement associated with the indeterminacy source.

Constraints

[1] *IndeterminacySpecification Constraint:* The *Element* associated with IndeterminacySpecification by *triggeredBy* must apply «IndeterminacySourceInput».

```
self.triggeredBy.getAppliedStereotype('UUP:IndeterminacySoureInput')<>null
```

2.1.3.3 EnablePattern

This enumeration is implemented to define options of pattern to enable indeterminacy source.

Literals

- Random

  The indeterminacy source is introduced during execution by random

- Always

  The indeterminacy source is always introduced during execution
- Measured

  The indeterminacy source is introduced during execution by specified measurement, e.g. it can be normal distribution
- Never

  The indeterminacy source is never introduced during execution

2.1.3.4 FindPosition

This enumeration is implemented to define options of position to enable indeterminacy source.

Literals

- Random

  The position of introducing indeterminacy source is random
- Any_Previous

  The position of introducing indeterminacy source located in any previous position before arrive the associated uncertainty
- Just_Previous

  The position of introducing indeterminacy source located in one previous position before arrive the associated uncertainty
- Specified

  The position of introducing indeterminacy source located in the specified position modeled in the test ready model by IndetermiancySourceInput

2.1.3.5 SelectSpecification

This enumeration is implemented to define options of specification to enable indeterminacy source.

Literals

- All

  All associated IndeterminacySpecification should be introduced
- Random

  Randomly select the specification are enabled during test execution
- Specified

  the indeterminacy specifications are enabled during test execution based on the specified ones by “enabled” attributes

2.1.4 Measurement Profile

This section presents the UML profile for the measurement. The corresponding profile diagram is shown in Figure 5.

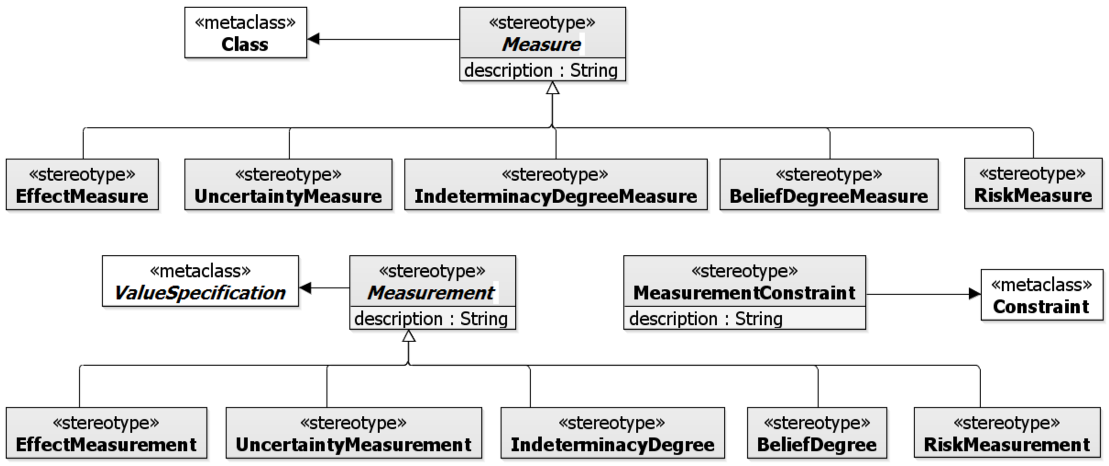

Figure 5. UML profile diagram for Measurement modeling

2.1.4.1 «BeliefDegreeMeasure»

This stereotype implements the Measure concept of a BeliefDegree as described in U-Model::BeliefModel [1].

Extensions

- None

Generalizations

- «Measure»

Associations

- None

Attributes

- None

Constraints

- None

2.1.4.2 «BeliefDegree»

This stereotype implements the Measurement concept of a BeliefElement as described in U-Model::BeliefModel [1].

Extensions

- None

Generalizations

- Measurement

Associations

- None

Attributes

- None

Constraints

- None

2.1.4.3 «EffectMeasure»

This stereotype implements the Measure concept of an Effect as described in U-Model::UncertaintyModel [1].

Extensions

- None

Generalizations

- «Measure»

Associations

- None

Attributes

- None

Constraints

- None

2.1.4.4 «EffectMeasurement»

This stereotype implements the Measurement concept of an Effect as described in U-Model::UncertaintyModel [1].

Extensions

- None

Generalizations

- «Measurement»

Associations

- None

Attributes

- None

Constraints

- None

2.1.4.5 «IndeterminacyDegreeMeasure»

This stereotype implements the Measure concept of an IndeterminacyDegree as described in U-Model::BeliefModel [1].

Extensions

- None

Generalizations

- «Measure»

Associations

- None

Attributes

- None

Constraints

- None

2.1.4.6 «IndeterminacyDegree»

This concept implements the Measurement concept of an IndeterminacySource as described in U-Model::BeliefModel [1].

Extensions

- None

Generalizations

- «Measurement»

Associations

- None

Attributes

- None

Constraints

- None

2.1.4.7 «Measure»

This stereotype implements the semantic of the U-Model::MeasureModel::Measure concept [1].

Extensions

- uml::Class
- uml::DataType

Generalizations

- None

Associations

- None

Attributes

- description: String[0..1]

  An optional description of a measure as a string

Constraints

- None

2.1.4.8 «Measurement»

This stereotype implements the semantic of the U-Model::BeliefModel::Measurement concept [1].

Extensions

- uml::Package
- uml::ValueSpecification

Generalizations

- None

Associations

- None

Attributes

- description: String[0..1]

  An optional description of a measurement as a string.

Constraints

- None

2.1.4.9 «RiskMeasure»

This stereotype implements Measure of a Risk as described in U-Model::UncertaintyModel [1].

Extensions

- None

Generalizations

- «Measure»

Associations

- None

Attributes

- None

Constraints

- None

2.1.4.10 «RiskMeasurement»

This stereotype implements the Measurement of a Risk as described in U-Model::UncertaintyModel [1].

Extensions

- None

Generalizations

- «Measurement»

Associations

- None

Attributes

- None

Constraints

- None

2.1.4.11 «UncertaintyMeasure»

This stereotype implements the Measure concept of an Uncertainty as described in U-Model::BeliefModel [1].

Extensions

- None

Generalizations

- Measure

Associations

- None

Attributes

- None

Constraints

- None

2.1.4.12 «UncertaintyMeasurement»

This stereotype implements the Measurement concept of an Uncertainty as described in U-Model::BeliefModel [1].

Extensions

- None

Generalizations

- «Measurement»

Associations

- None

Attributes

- None

Constraints

- None

2.2 Internal *UUP* Library

This section presents our internal library that defines the various types used in the *UUP*. An overview of the diagram is shown in Figure 6 followed by description.

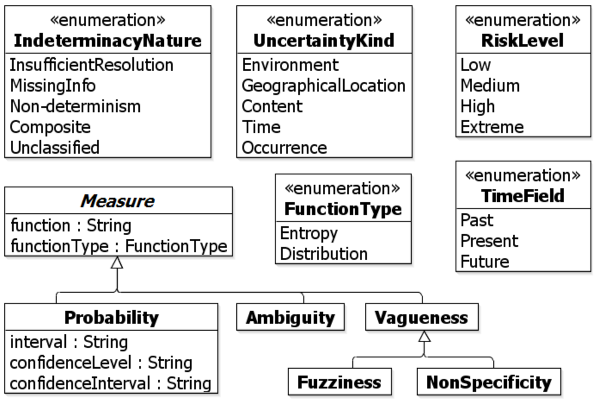

Figure 6. Internal Library

2.2.1 Ambiguity

This concept implements the semantics from U-Model::MeasureModel::Ambiguity [1].

Extensions

- None

Generalizations

- None

Associations

- None

Attributes

- None

Constraints

- None

2.2.2 FunctionType

This enumeration defines the type of function as defined in the package of U-Model::MeasureModel [1].

Literals

- Entropy

  As defined in U-Model::MeasureModel::Entropy [1]

- Distribution

  As defined in U-Model::MeasureModel::Distribution [1]

2.2.3 Measure

This concept implements the semantics of the U-Model::BeliefModel::Measure concept [1].

Extensions

- None

Generalizations

- None

Associations

- None

Attributes

- function: String[0..1]

  Optional specification of the function to use as a measure

- functionType: FunctionType[0..1]

  Optional specification of the type of function as one of the literals from the FunctionType enumeration.

Constraints

- None

2.2.4 NonSpecificity

Semantics are adopted from U-Model::MeasureModel:: NonSpecificity [1]

Extensions

- None

Generalizations

- Vagueness

Associations

- None

Attributes

- None

Constraints

- None

2.2.5 IndeterminacyNature

Semantics adopted from U-Model::BeliefModel::IndeterminacyNature [1].

Literals

- InsufficientResolution

  As defined in U-Model::BeliefModel::IndeterminacyNature:: InsufficientResolution [1]

- MissingInfo

As defined in U-Model::BeliefModel::IndeterminacyNature::MissingInfo [1]

- Non-determinism

  As defined in U-Model::BeliefModel::IndeterminacyNature::Non-determinism [1]
- Composite

  As defined in U-Model::BeliefModel::IndeterminacyNature::Composite [1]
- Unclassified

  As defined in U-Model::BeliefModel::IndeterminacyNature::Unclassified [1]

2.2.6 Probability

Semantics from U-Model::MeasureModel::Probabilitys [1]

Extensions

- None

Generalizations

- Measure

Associations

- None

Attributes

- interval: String[0..1]

  Optional specification of an interval of probability as a String
- confidenceLevel: String[0..1]

  Optional specification of the confidence level related to Probability
- confidenceInterval: String[0..1]

  Optional specification of the confidence interval related to Probability

Constraints

- None

2.2.7 RiskLevel

Semantics from U-Model::UncertaintyModel::RiskLevel [1]

Literals

- Low

  Specifying the lowest level of risk
- Medium

  Specifying the medium level of the risk
- High

  Specifying the high level of the risk
- Extreme

  Specifying the highest level of the risk

2.2.8 TimeField

Semantics from U-Model::UncertaintyModel::TimeField [1]

Literals

- Past

  As defined in U-Model::TimeField::Past [1]
- Present

  As defined in U-Model::TimeField::Present [1]
- Future

  As defined in U-Model::TimeField::Future [1]

2.2.9 UncertaintyKind

This enumeration defined the type of uncertainty as defined in the package of U-Model::UncertaintyModel [1]

Literals

- Environment

  As defined in U-Model::UncertaintyModel::Environment [1]
- GeographicalLocation

  As defined in U-Model::UncertaintyModel::GeographicalLocation [1]
- Content

  As defined in U-Model::UncertaintyModel::Content [1]
- Time

  As defined in U-Model::UncertaintyModel::Time [1]
- Occurrence

  As defined in U-Model::UncertaintyModel::Occurrence [1]

2.2.10 Vagueness

Semantics from U-Model::MeasureModel::Vagueness [1]

Extensions

- None

Generalizations

- Measure

Associations

- None

Attributes

- None

Constraints

- None

2.3 CPS Profile

This section presents the CPS Profile to denote which model element belongs to which of the three CPS testing levels: *Application*, *Infrastructure* and *Integration*.

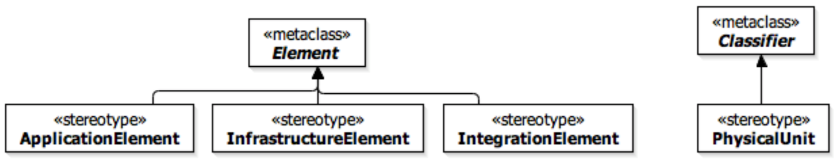

Figure 7. CPS Profile

3 MODEL LIBRARIES

This section presents the four model libraries that we defined including: 1) Measure Library, 2) Pattern Library, 3) Risk Library, 4) Time Library.

3.1 Measure Library

The *Measure* library consists of three further libraries including: 1) Ambiguity, 2) Probability, and 3) Vagueness libraries.

3.1.1 Ambiguity Library

In this section, we describe different measures that are related to the Ambiguity [1]. An overview diagram is shown in Figure 8.

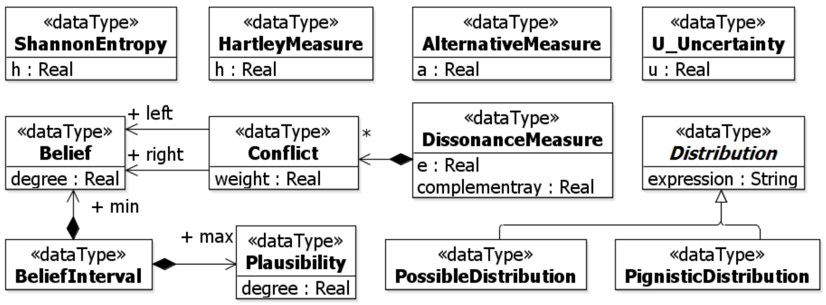

Figure 8. Ambiguity Library

3.1.1.1 BeliefInterval

This data type contains necessary parameters to define a belief interval[1] [3].

Associations

- min: Belief

  the lower bound of BeliefInterval specified by Belief
- max: Plausibility

  the upper bound of BeliefInterval specified by Plausibility

3.1.1.2 Belief

This data type contains necessary parameters to define a belief[2] [3].

Attributes

- degree: Real

  the value of degree of belief

3.1.1.3 Plausibility

This data type contains necessary parameters to define a plausibility[3] [3].

Attributes

- degree: Real

  the value of degree of plausibility

3.1.1.4 Distribution

This abstract data type implements to specify the common known distribution in the mathematics.

Attributes

- expression: String [1]

  the expression of the Distribution specified by String

3.1.1.5 PossibleDistribution

This data type implements to specify the possible distribution [4].

3.1.1.6 PignisticDistribution

This data type implements to specify the pignistic distribution [5].

3.1.1.7 HartleyMeasure

This data contains the necessary parameter to specify value using Hartley measure [6] to measure uncertainty.

Attributes

- h: Real [1]

  the value specified by Hartley measure

[1] This concept borrows from *Dempster-Shafer* is used to specify belief interval that is a boundary of probability.
[2] This concept from belief function in *Dempster-Shafer Theory* represents the degree of belief supported by the evidence directly.
[3] This concept from plausibility function in *Dempster-Shafer Theory* represents the degree of plausibility supported by maximum share of the evidence.

3.1.1.8 U_Uncertainty

This data type contains the necessary parameter to specify value using U-uncertainty function [7] to measure uncertainty.

Attributes

- u: Real [1]

  the value specified by U-uncertainty

3.1.1.9 AlternativeMeasure

This data type contains the necessary parameter to specify value using the alternative measure [8] to measure uncertainty.

Attributes

- a: Real [1]

  the value specified by alternative measure

3.1.1.10 ShannonEntropy

This data type contains the necessary parameter to specify value using Shannon Entropy to measure uncertainty.

Attributes

- h: Real [1]

  the value of Shannon entropy with default bit unit

3.1.1.11 Conflict

This data type is used to specify the weight of conflict between two beliefs [9].

Associations

- weight: Real[1]

  the value of weight of conflict between beliefs
- left: Belief [0..1]

  the first parameter of conflict function [9]
- right: Belief [0..1]

  the second parameter of conflict function [9]

3.1.1.12 DissonanceMeasure

This data type contains the necessary parameter to specify value using dissonance measure [10] to measure uncertainty.

Attributes

- e: Real [1]

  the value specified by Dissonance measure
- complementary: Real [0..1]

  optional value of complementary [11, 12]

Associations

- conflict: Conflict [*]

  the set of conflicts used for computing the value specified by Dissonance measure

3.1.2 Probability Library

This section presents our Probability Library. An overview diagram is shown in Figure 9.

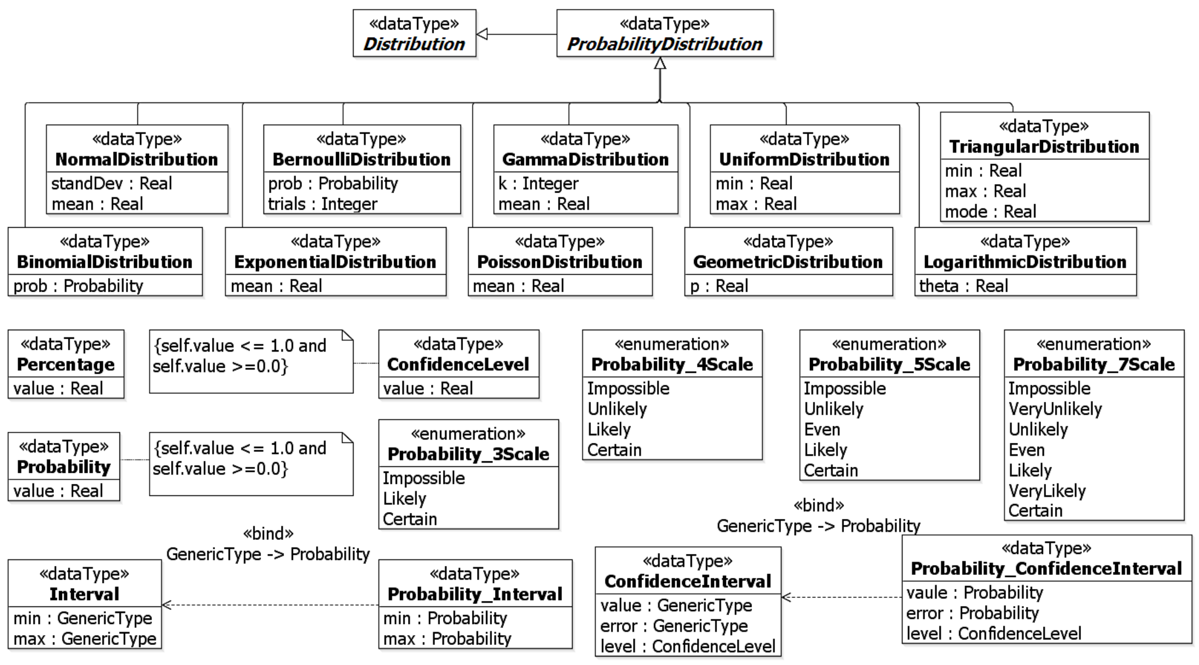
«dataType» Distribution
«dataType» ProbabilityDistribution
«dataType» NormalDistribution
standDev : Real
mean : Real
«dataType» BernoulliDistribution
prob : Probability
trials : Integer
«dataType» GammaDistribution
k : Integer
mean : Real
«dataType» UniformDistribution
min : Real
max : Real
«dataType» TriangularDistribution
min : Real
max : Real
mode : Real
«dataType» BinomialDistribution
prob : Probability
«dataType» ExponentialDistribution
mean : Real
«dataType» PoissonDistribution
mean : Real
«dataType» GeometricDistribution
p : Real
«dataType» LogarithmicDistribution
theta : Real
«dataType» Percentage
value : Real
{self.value <= 1.0 and self.value >=0.0}
«dataType» ConfidenceLevel
value : Real
«enumeration» Probability_4Scale
Impossible
Unlikely
Likely
Certain
«enumeration» Probability_5Scale
Impossible
Unlikely
Even
Likely
Certain
«enumeration» Probability_7Scale
Impossible
VeryUnlikely
Unlikely
Even
Likely
VeryLikely
Certain
«dataType» Probability
value : Real
{self.value <= 1.0 and self.value >=0.0}
«enumeration» Probability_3Scale
Impossible
Likely
Certain
«bind»
GenericType -> Probability
«bind»
GenericType -> Probability
«dataType» Interval
min : GenericType
max : GenericType
«dataType» Probability_Interval
min : Probability
max : Probability
«dataType» ConfidenceInterval
value : GenericType
error : GenericType
level : ConfidenceLevel
«dataType» Probability_ConfidenceInterval
vaule : Probability
error : Probability
level : ConfidenceLevel

Figure 9. Probability Library

3.1.2.1 ProbabilityDistribution

This abstract concept represents the common probability distribution concept, and we further implement 10 common known probability distributions in the probability model library.

3.1.2.2 NormalDistribution

This data type contains necessary parameters to define the Normal distribution.

Attributes

- standDev: Real [1]

  the value of standard deviation in the normal distribution
- mean: Real [1]

  the value of mean in the normal distribution

3.1.2.3 BinomialDistribution

This data type contains necessary parameters to define the Binomial distribution.

Attributes

- prob: Probability [1]

  the probability in the binomial distribution

3.1.2.4 BernoulliDistribution

This data type contains necessary parameters to define the Bernoulli distribution.

Attributes

- prob: Probability [1]

  the probability in the Bernoulli distribution
- trials: Integer [1]

  the trials in the Bernoulli distribution

3.1.2.5 ExponentialDistribution

This data type contains necessary parameters to define the Exponential distribution.

Attributes

- mean: Real [1]

  the value of mean in the exponential distribution

3.1.2.6 GammaDistribution

This data type contains necessary parameters to define the Gamma distribution.

Attributes

- k: Integer [1]

  the value of shape parameter in the gamma distribution
- mean: Real [1]

  the value of mean (scale parameter) in the gamma distribution

3.1.2.7 PoissonDistribution

This data type contains necessary parameters to define the Poisson distribution.

Attributes

- mean: Real [1]

  the value of mean in the passion distribution

3.1.2.8 UniformDistribution

This data type contains necessary parameters to define the Uniform distribution.

Attributes

- min: Real [1]

  the value of minimum in the uniform distribution
- max: Real [1]

  the value of maximum in the uniform distribution

3.1.2.9 GeometricDistribution

This data type contains necessary parameter to define the Geometric distribution.

Attributes

- p: Real [1]

  the value of parameter p in the Geometric distribution

3.1.2.10 TriangularDistribution

This data type contains necessary parameters to define the Triangular distribution.

Attributes

- min: Real [1]

  the value of lower limit, normally represents as *a* in the triangular distribution
- max: Real [1]

  the value of upper limit, normally represents as *b* in the triangular distribution
- mode: Real [1]

  the value of mode, normally represents as *c* in the triangular distribution

Constraints

[1] min is less than max

```
self.min < self.max
```

[2] mode is between min and max

```
self.min <= self.mode and self.max >= self.mode
```

3.1.2.11 LogarithmicDistribution

This data type contains necessary parameters to define the Logarithmic distribution.

Attributes

- theta: Real

  the value of parameter theta in the Logarithmic distribution

3.1.2.12 Percentage

This data type implements the common known percentage concept.

Attributes

- value: Real

  the value of percentage with default % unit

3.1.2.13 Probability

This data type contains necessary parameter to specify the common probability value.

Attributes

- value: Real

  the value of probability

Constraints

[1] A probability value ranges from 0.0 to 1.0

```
self.value <= 1.0 and self.value > 0.0
```

3.1.2.14 ConfidenceLevel

This data type contains necessary parameters to specify the value of confidence level.

Attributes

- value: Probability [1]

  the value of confidence level specified by Probability

3.1.2.15 ConfidenceInterval

This data type contains necessary parameters to specify the value of confidence interval.

Attributes

- value: GenericType [1]

  the value of estimator

- error: GenericType [1]

  the value of error in the confidence interval

- level: ConfidenceLevel [1]

  the value of confidence level, also called confidence coefficient, in the confidence interval

3.1.3 Vagueness Library

This section presents the Vagueness Library [1]. An overview diagram is shown in Figure 10.

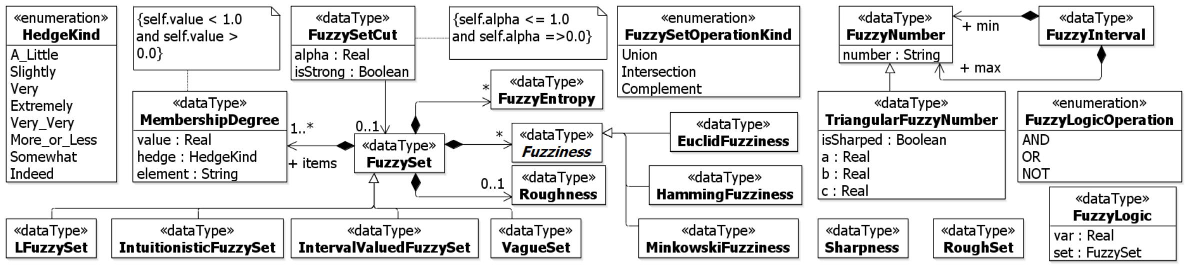

Figure 10. Vagueness Library

3.1.3.1 HedgeKind

This enumeration implements the classical kind of Hedge related Fuzzy Set [13].

Literals

- A_Little

  corresponds to the mathematic representation $[degree\ of\ memebership]^{1.3}$

- Slightly

  corresponds to the mathematic representation $[degree\ of\ memebership]^{1.7}$

- Very

  corresponds to the mathematic representation $[degree\ of\ memebership]^{2}$

- Extermely

  corresponds to the mathematic representation $[degree\ of\ memebership]^{3}$

- Very_Very

  corresponds to the mathematic representation $[degree\ of\ memebership]^{4}$

- More_or_Less

corresponds to the mathematic representation $\sqrt{[degree\ of\ membership]}$

- Somewhat

  corresponds to the mathematic representation $\sqrt{[degree\ of\ membership]}$

- Indeed

  corresponds to the mathematic representation

$$\begin{cases} 2[degree\ of\ memebership]^2 & ,\ 0 \leq degree\ of\ membership \leq 0.5 \\ 1-2[degree\ of\ memebership]^2 & ,\ 0.5 < degree\ of\ memebership \leq 1 \end{cases}$$

3.1.3.2 MembershipDegree

This data type contains necessary parameters to specify the value of membership degree.

Attributes

- value: Real [1]

  the value of membership

- hedge: HedgeKind [0..1]

  optional value to specify the kind of hedge

- element: String [0..1]

  the value to specify the element

3.1.3.3 FuzzySet

This data type contains necessary parameters to specify the value of fuzzy set [14].

Associations

- items: MemebershipDegree [*]

  the set of membership in the fuzzy set

- fuzzyEntropy: FuzzyEntropy [*]

  the set of fuzzy entropy of the fuzzy set

- fuzziness: Fuzziness[*]

  the set of the degree of fuzziness of the fuzzy set

- roughness: Roughness[0..1]

  optional value of the degree of roughness of the fuzzy set

3.1.3.4 FuzzySetCut

This data type contains necessary parameters to specify the cut of fuzzy set [15].

Attributes

- alpha: Real [1]

  the value of alpha in the fuzzy cut

- isStrong: Boolean [0..1]

  the value to represent the strong fuzzy cut or not

Associations

- fuzzySet: FuzzySet [1]

  the value of fuzzy set for the cut

Constraints

[1] A alpha value ranges from 0.0 to 1.0

```
self.alpha <= 1.0 and self.alpha =>0.0
```

3.1.3.5 FuzzyEntropy

This data type implements to specify the value of fuzzy entropy [16].

3.1.3.6 Fuzziness

This data type implements to specify the value of fuzziness [17, 18].

3.1.3.7 Roughness

This data type implements to specify the value of roughness [19].

3.1.3.8 EuclidFuzziness
This data type implements to specify the value of Euclid fuzziness [17, 18].
Generalizations
- Fuzziness

3.1.3.9 HammingFuzziness
This data type implements to specify the value of Hamming fuzziness [17, 18].
Generalizations
- Fuzziness

3.1.3.10 MinkowskiFuzziness
This data type implements to specify the value of Minkowski fuzziness [17, 18].
Generalizations
- Fuzziness

3.1.3.11 FuzzySetOperationKind
This enumeration implements the common kind of operation of fuzzy set.
Literals
- Union

  Union operation between fuzzy set
- Intersection

  Intersection operation between fuzzy set
- Complement

  Complement operation between fuzzy set

3.1.3.12 FuzzyLogicOperation
This enumeration implements the common kind of operation supported by fuzzy logic.
Literals
- AND

  And operation
- OR

  Or operation
- NOT

  Not operation

3.1.3.13 LFuzzySet
This data type implements to specify the value of L-Fuzzy Set [20].
Generalizations
- FuzzySet

3.1.3.14 IntuitionisticFuzzySet
This data type implements to specify the value of Intuitionistic fuzzy set [21].
Generalizations
- FuzzySet

3.1.3.15 IntervalValuedFuzzySet
This data type implements to specify the value of interval valued fuzzy set [22-24].
Generalizations
- FuzzySet

3.1.3.16 VagueSet
This data type implements to specify the value of vague set [25].
Generalizations
- FuzzySet

3.1.3.17 FuzzyLogic
This data type implements to specify the fuzzy logic [14].

Attributes

- var: String [1]

  the variable in the fuzzy logic
- set: FuzzySet [1]

  the fuzzy set used for mapping the value

3.1.3.18 FuzzyNumber
This data type implements to specify the fuzzy number [14].

Attributes

- number: String [1]

  the value of fuzzy number specified by String

3.1.3.19 TriangularFuzzyNumber
This data type implements the necessary parameters to specify the triangular fuzzy number.

Generalizations

- FuzzyNumber

Attributes

- isSharped: Boolean[1]

  the value to decide triangular fuzzy number or triangular sharped fuzzy number
- a: Real [1]

  the lower bound of triangular fuzzy number
- b: Real [1]

  the medium bound of triangular fuzzy number
- c: Real [1]

  the upper bound of triangular fuzzy number

3.1.3.20 FuzzyInterval
This data type implements to specify the fuzzy interval [14].

Attributes

- min: FuzzyNumber[1]

  the lower bound of fuzzy interval
- max: FuzzyNumber [1]

  the upper bound of fuzzy interval

3.1.3.21 RoughSet
This data type implements to specify the rough set [19].

3.1.3.22 Sharpness
This data type implements to specify the value of sharpness [26].

3.2 Pattern Library
This section presents the Pattern Library [1]. An overview diagram is shown in Figure 11.

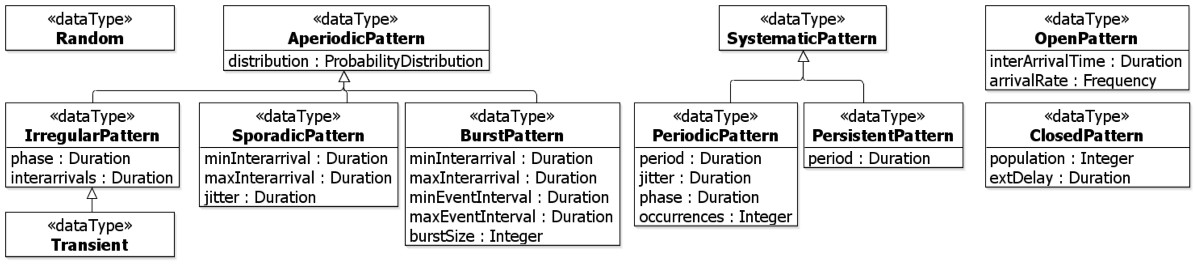
«dataType» Random
«dataType» AperiodicPattern
distribution : ProbabilityDistribution
«dataType» SystematicPattern
«dataType» OpenPattern
interArrivalTime : Duration
arrivalRate : Frequency
«dataType» IrregularPattern
phase : Duration
interarrivals : Duration
«dataType» SporadicPattern
minInterarrival : Duration
maxInterarrival : Duration
jitter : Duration
«dataType» BurstPattern
minInterarrival : Duration
maxInterarrival : Duration
minEventInterval : Duration
maxEventInterval : Duration
burstSize : Integer
«dataType» PeriodicPattern
period : Duration
jitter : Duration
phase : Duration
occurrences : Integer
«dataType» PersistentPattern
period : Duration
«dataType» ClosedPattern
population : Integer
extDelay : Duration
«dataType» Transient

Figure 11. Pattern Library

3.2.1.1 Random

This data type implements the Random in the U-Model::UncertaintyModel::Random [1]. As defined in [1], "An Random Uncertainty that occurs without definite method, purpose or conscious decision".

3.2.1.2 AperiodicPattern

This data type implements the Aperiodic in the U-Model::UncertaintyModel::Aperiodic [1], and borrow the parameters from MARTE Library [2]. As defined in [1], "an Aperiodic pattern occurs at irregular intervals of time".

Attributes

- distribution: ProbabilityDistribution

  As defined in the MARTE_Library:: BasicNFP_Types::AperiodicPattern [2]

3.2.1.3 IrregularPattern

This data type from MARTE [2] contains the necessary parameters to specify irregular pattern.

Generalizations

- AperiodicPattern

Attributes

- phase: Duration

  As defined in the MARTE_Library:: BasicNFP_Types::IrregularPattern [2]
- interarrivals: Duration

  As defined in the MARTE_Library:: BasicNFP_Types::IrregularPattern [2]

3.2.1.4 SporadicPattern

This data type implements the Sporadic in the U-Model::UncertaintyModel::Sporadic [1], and borrow the parameters from MARTE Library [2]. As defined in [1], "an uncertainty occurs occasionally".

Generalizations

- AperiodicPattern

Attributes

- minInterarrival: Duration

  As defined in the MARTE_Library:: BasicNFP_Types::SporadicPattern [2]
- maxInterarrival: Duration

  As defined in the MARTE_Library:: BasicNFP_Types::SporadicPattern [2]
- jitter: Duration

  As defined in the MARTE_Library:: BasicNFP_Types::SporadicPattern [2]

3.2.1.5 BurstPattern

This data type from MARTE [2] contains the necessary parameters to specify burst pattern.

Generalizations

- AperiodicPattern

Attributes

- minInterarrival: Duration

  As defined in the MARTE_Library:: BasicNFP_Types::BurstPattern [2]
- maxInterarrival: Duration

  As defined in the MARTE_Library:: BasicNFP_Types::BurstPattern [2]
- miniEventInterval: Duration

  As defined in the MARTE_Library:: BasicNFP_Types::BurstPattern [2]
- maxEventInterval: Duration

  As defined in the MARTE_Library:: BasicNFP_Types::BurstPattern [2]
- burstSize: Integer

  As defined in the MARTE_Library:: BasicNFP_Types::BurstPattern [2]

3.2.1.6 SystematicPattern

This data type implements the Systematic in the U-Model::UncertaintyModel:: Systematic [1]. As defined in [1], "an Systematic uncertainty occurs in some methodical pattern".

3.2.1.7 PeriodicPattern

This data type implements the Periodic in the U-Model::UncertaintyModel::Periodic [1], and borrow the parameters from MARTE Library [2]. As defined in [1], "an Uncertainty that occurs in repeated periods or at regular intervals.".

Generalizations

- SystematicPattern

Attributes

- period: Duration

  As defined in the MARTE_Library:: BasicNFP_Types::PeriodicPattern [2]
- jitter: Duration

  As defined in the MARTE_Library:: BasicNFP_Types::PeriodicPattern [2]
- phase: Duration

  As defined in the MARTE_Library:: BasicNFP_Types::PeriodicPattern [2]
- occurrences: Integer

  As defined in the MARTE_Library:: BasicNFP_Types::PeriodicPattern [2]

3.2.1.8 PersistentPattern

This data type implements the Persistent in the U-Model::UncertaintyModel:: Persistent [1]. As defined in [1], "an uncertainty lasts forever[4]".

Generalizations

- SystematicPattern

Attributes

- phase: Duration

  the duration of the uncertainty lasting

3.2.1.9 TransientPattern

This data type implements the Transientin the U-Model::UncertaintyModel::Transient[1]. As defined in [1], "an uncertainty does not last long".

Generalizations

- AperiodicPattern

3.2.1.10 OpenPattern

This data type from MARTE [2] contains the necessary parameters to specify open pattern.

Attributes

- interArrivalTime: Duration

  As defined in the MARTE_Library:: BasicNFP_Types::OpenPattern [2]
- arrivalRate: Frequency

  As defined in the MARTE_Library:: BasicNFP_Types::OpenPattern [2]

3.2.1.11 ClosePattern

This data type from MARTE [2] contains the necessary parameters to specify close pattern.

Attributes

- population: Integer

  As defined in the MARTE_Library:: BasicNFP_Types::ClosePattern [2]
- extDelay: Duration

[4] "The definition of "forever" varies. For example, an uncertainty may exist permanently until appropriate actions are taken to deal with the uncertainty. On the other hand, an uncertainty may not be able to resolve and stays forever." [1] M. Zhang, B. Selic, S. Ali, T. Yue, O. Okariz, and R. Norgren, "Understanding Uncertainty in Cyber-Physical Systems: A Conceptual Model," in ECMFA, 2016.

As defined in the MARTE_Library:: BasicNFP_Types::ClosePattern [2]

3.3 Time Library

This section presents time concepts borrowed from MARTE Library[2]. An overview diagram is shown in Figure 12.

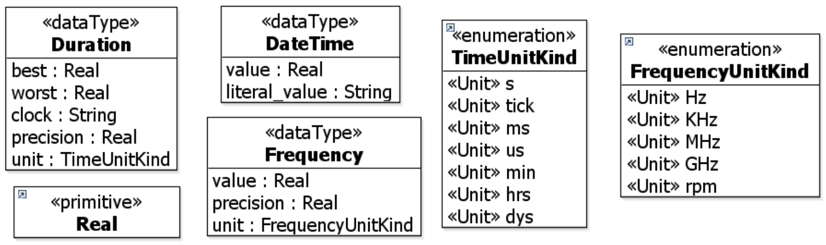

Figure 12. Time Library

3.3.1.1 DateTime

This data type from MARTE_Library [2] contains the necessary parameters to specify common date time.

Attributes

- value: Real [0..1]

  the data time specified by Real number
- literal_value: String [0..1]

  the date time specified by String

3.3.1.2 Duration

This data type from MARTE_Library [2] contains the necessary parameters to specify common duration.

Attributes

- best: Real [0..1]

  As defined in the MARTE_Library::BasicNFP_Types::NFP_Duration [2]
- worst: Real [0..1]

  As defined in the MARTE_Library::BasicNFP_Types::NFP_Duration [2]
- unit: MARTE_Library::MARTE_MeasurementUnits::TimeUnitKind [0..1]

  As defined in the MARTE_Library::BasicNFP_Types::NFP_Duration [2]
- clock: String [0..1]

  As defined in the MARTE_Library::BasicNFP_Types::NFP_Duration [2]
- precision: Real [0..1]

  As defined in the MARTE_Library::BasicNFP_Types::NFP_Duration [2]

3.3.1.3 Frequency

This data type from MARTE_Library [2] contains the necessary parameters to specify common frequencey.

Attributes

- value: Real [0..1]

  As defined in the MARTE_Library:: BasicNFP_Types::NFP_Frequency [2]
- precision: Real [0..1]

  As defined in the MARTE_Library:: BasicNFP_Types:: NFP_Frequency [2]
- unit: MARTE_Library::MARTE_MeasurementUnits::FrequencyUnitKind [0..1]

  As defined in the MARTE_Library:: BasicNFP_Types:: NFP_Frequency [2]

3.3.1.4 FrequencyUnitKind

This enumeration from MARTE_Library::MARTE_MeasurementUnits::FrequencyUnitKind [2] lists the kinds of unit of frequency.

literals

- Hz

  Represents *hertz*, one cycle per second

- KHz

  $kilohertz,\ 10^3\ Hz$
- MHz

  $megahertz, 10^6\ Hz$
- GHz

  $glgahertz, 10^9\ Hz$
- rmp

  radian per second, $0.0167Hz, 2\pi\ rmp = 1\ Hz$

3.3.1.5 TimeUnitKind

This enumeration from MARTE_Library::MARTE_MeasurementUnits::TimeUnitKind [2] lists the kinds of unit of time.

literals

- s

  second
- tick

  represents logic time, often used for logic clock
- ms

  millisecond $10^{-3}\ s$
- us

  microsecond $10^{-3}\ ms$
- min

  minute, $60\ s$
- hrs

  hour, $60\ min$
- dys

  day, $24\ hrs$

3.4 Risk Library

This section describes our model library for risk assessment that consists of some key approaches for assessing risk. Figure 13 shows an overview of our model library for risk assessment. This model library allows us to specify the key elements that are used for assessing risk, either quantitatively or qualitatively. Therefore, the library contains two sub model libraries *RiskQualitativeAssessment* and *RiskQuantitativeAssessment*. Besides, some data types are imported from the Probability model library.

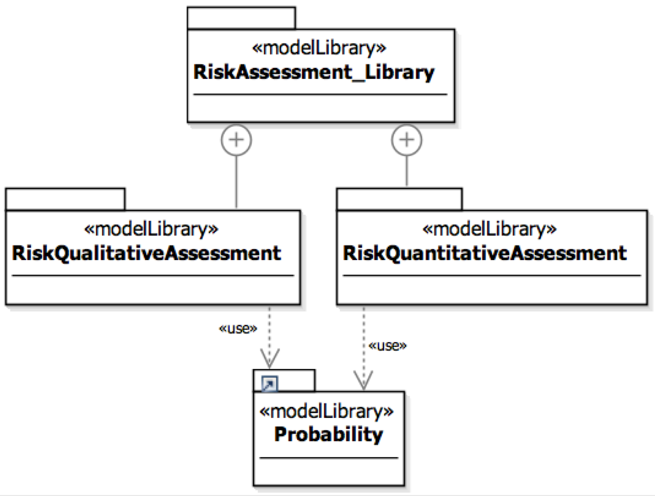

Figure 13. An overview of Risk Assessment model library

3.4.1 Risk Qualitative Assessment

To qualitatively assess risk, we adopt the *Risk Matrix* approach [27] for identifying, assessing, and ranking risks. According to [28], to produce a risk matrix, some basic rules should be followed. First, the basis for risk matrix is the standard definition of risk as a combination of severity of the consequences occurring in a certain accident scenario and its probability [29]. Second, the severity of consequences, probability, and output risk index can be divided into different levels, respectively, with qualitative descriptions and scales. Third, the calculation process of matrix producing is presented by the logic implication as: IF probability is P AND severity of consequence (impact) is I THEN risk is R [29]. Some examples of impact assessment, probability of occurrence, and a possible Risk Matrix (risk rating scale) is recalled from [27] in Table 1, Table 2, and Table 3.

Table 1. Probability of Occurrence (P)

Probability Range	Interpretation
0-10%	Very Unlikely to Occur
11-40%	Unlikely to Occur
41-60%	May Occur About Half of the Time
61-90%	Likely to Occur
91-100%	Very Likely to Occur

Table 2. Risk Matrix Impact Assessment (I)

Impact Category	Definition
Critical	An event that, if it occurred, would cause program failure (inability to achieve minimum acceptable requirements).
Serious	An event that, if it occurred, would cause major cost/schedule increases. Secondary requirements may not be achieved.
Moderate	An event that, if it occurred, would cause moderate cost/schedule increases. but important requirements would still be met.
Minor	An event that, if it occurred, would have cause only a small cost/schedule increase. Requirements would still be achieved.
Negligible	An event that, if it occurred, would have no effect on the program.

Table 3. Possible Risk Rating Scale (R)

P \ I	Negligible	Minor	Moderate	Serious	Critical
0-10%	Low	Low	Low	Medium	Medium
11-40%	Low	Low	Medium	Medium	High
41-60%	Low	Medium	Medium	Medium	High
61-90%	Medium	Medium	Medium	Medium	High
91-100%	Medium	High	High	High	High

In the model library we introduce data types such as RiskLevel, Risk_ProbabilityLevel, and Risk_ImpactLevel as showed in Figure 14.

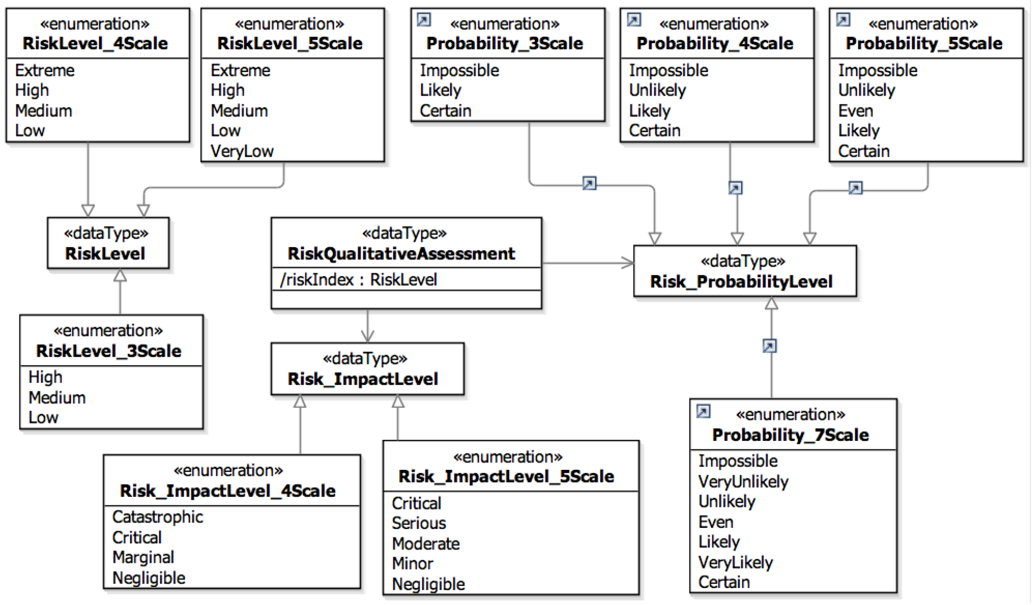
«enumeration» RiskLevel_4Scale
Extreme
High
Medium
Low
«enumeration» RiskLevel_5Scale
Extreme
High
Medium
Low
VeryLow
«enumeration» Probability_3Scale
Impossible
Likely
Certain
«enumeration» Probability_4Scale
Impossible
Unlikely
Likely
Certain
«enumeration» Probability_5Scale
Impossible
Unlikely
Even
Likely
Certain
«dataType» RiskLevel
«dataType» RiskQualitativeAssessment
/riskIndex : RiskLevel
«dataType» Risk_ProbabilityLevel
«enumeration» RiskLevel_3Scale
High
Medium
Low
«dataType» Risk_ImpactLevel
«enumeration» Probability_7Scale
Impossible
VeryUnlikely
Unlikely
Even
Likely
VeryLikely
Certain
«enumeration» Risk_ImpactLevel_4Scale
Catastrophic
Critical
Marginal
Negligible
«enumeration» Risk_ImpactLevel_5Scale
Critical
Serious
Moderate
Minor
Negligible

Figure 14. Using Risk Matrix for Qualitative Assessment of Risk

The value riskIndex in RiskQualitativeAssessment can be derived from the values of *Risk*_ProbabilityLevel and *Risk*_ImpactLevel according to the calculation process of matrix producing. Each data type is presented in the following.

3.4.1.1 RiskQualtitativeAssessment

This data type contains the derived risk index.

Extensions

- None

Generalizations

- None

Associations

- impact : Risk_ImpactLevel [1]

  Any risk is based on impact assessment. This impact is assigned a value of type Risk_ImpactLevel below.

- probability : Risk_ProbabilityLevel [1]

  Any risk is based on a probability of its occurrence. This probability is assigned a value of type Risk_ProbabilityLevel below.

Attributes

- /riskIndex : RiskLevel [1]

  This attribute represents the qualitative assessment of risk.

Constraints

- None

3.4.1.2 RiskLevel

This data type represents the possible risk rating scale.

Extensions

- None

Generalizations

- None

Associations

- None

Attributes

- None

Constraints

- None

3.4.1.3 RiskLevel_3Scale, RiskLevel_4Scale, RiskLevel_5Scale

RiskLevel_3Scale, RiskLevel_4Scale, and RiskLevel_5Scale are enumeration types that define literals used to specify the levels of risk. We provide different scales to be used in different situations. In this version, we have three different scales with the corresponding levels. The indication of each level can be defined in a Risk Matrix, e.g. the three-scale risk ranking in Table 3. A risk matrix can be created according to the need of granularity in risk assessment.

Literals

- VeryLow
- Low
- Medium
- High
- Extreme

Generalizations

- RiskLevel

3.4.1.4 Risk_ProbabilityLevel

This data type contains the possible risk probability rating scale.

Extensions

- None

Generalizations

- None

Associations

- None

Attributes

- None

Constraints

- None

3.4.1.5 Probability_3Scale, Probability_4Scale, Probability_5Scale, Probability_7Scale

Probability_3Scale, Probability_4Scale, Probability_5Scale, and Probability_5Scale are enumeration types that define literals used to specify the probability if a risk could occur. We provide different scales to be used in different situations. The enumerations are imported from the Probability model library. The indication of each level of impact can be defined in table, e.g. Table 1.

Literals

- Impossible
- VeryUnlikely
- Unlikely
- Even
- Likely
- VeryLikely
- Certain

Generalizations

- Risk_ProbabilityLevel

3.4.1.6 Risk_ImpactLevel

This data type contains the possible risk impact rating scale.

Extensions

- None

Generalizations

- None

Associations

- None

Attributes

- None

Constraints

- None

3.4.1.7 Risk_ImpactLevel_4Scale, Risk_ImpactLevel_5Scale

Risk_ImpactLevel_4Scale, Risk_ImpactLevel_5Scale are enumeration types that define literals used to specify the levels of risk impact. We provide different scales to be used in different situations. In this version, we introduce two different scales with the corresponding levels. The indication of each level of impact can be defined like in Table 2.

Literals

- VeryLow
- Low
- Medium
- High
- Extreme

Generalizations

- Risk_ImpactLevel

3.4.2 Risk Quantitative Assessment

To quantitatively assess risk, we adopt the extension of risk matrix approach [28]. In [2], the authors claim that increasing the number of input variable levels and risk levels will indirectly reduce the probability of risk tie occurrence. Moreover, in the limiting case, the levels of two input variables are single discrete values instead of several equal length value ranges. In this case, to maximize the elimination of risk ties, the original calculation process of Risk Matrix approach must be accordingly redefined in another way better than logic implication and Borda method [28].

The multiplication formula could naturally be selected as the new calculation process for the extension of Risk Matrix approach because the most commonly used way to express risk is the expected value. The calculation process of risk matrix can be redefined to quantify risk index as follows:

$$Risk = Severity \times Probability$$

Therefore, in the model library for quantitatively assessing risk we have data types such as Severity, Risk_QuantitativeAssessment as showed in Figure 15.

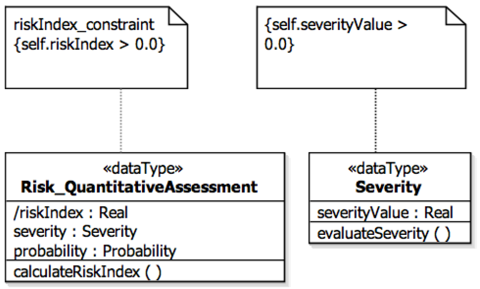

Figure 15. Overview of a model library for Quantitative Assessment of Risk

3.4.2.1 Risk_QuantitativeAssessment

This data type contains the quantifiable value of risk.

Extensions

- None

Generalizations

- None

Associations

- None

Attributes

- riskIndex : Real [1]

The value of risk index.

- severity : Severity [1]

  The severity of risk.
- probability : Probability [1]

  The probability of risk. This data type is imported from the Probability model library.

Constraints

- The value of riskIndex must be positive.

3.4.2.2 Severity

This data type contains the possible severity of consequences of risk.

Extensions

- None

Generalizations

- None

Associations

- None

Attributes

- severityValue : Real [1]

  The value of severity.

Constraints

- The value of severityValue must be positive.

4 Modeling Methodology

In this section, we present a modeling methodology for the *U-Model* notations. The rest of this section is organized as follows: Section 4.1 presents the overview of modeling activities, Section 4.2 presents modeling activities at *Application Level*, Section 4.3 presents modeling activities at *Infrastructure Level*, Section 4.4 presents modeling activities at *Integration level*, and Section 4.5 presents the modeling activities of *applying UUP* which is invoked at above three level.

4.1 Overview

The modeling methodology is naturally organized from the viewpoints of the three types of stakeholders: *Application* Modeler, *Infrastructure* Modeler and *Integration* Modeler, as shown in Figure 16. For activities performed by each type of modelers, we distinguish them by tagging each of them (in their names) using "AP", "IF" and "IT", respectively.

As shown in Figure 16, all modelers are recommended to start from creating a package (i.e., AP1, IF1 and IT1), which is used to group and contain model elements for each respective level. Next, application and infrastructure modelers apply the *U-Model* notations to model system behaviors of the application and infrastructure levels, respectively (i.e., AP2 and IF2). These two structured activities are further elaborated in Sections 4.2 and 4.3. When these two activities are finished, integration modelers take their results as inputs and perform *IT2: Model Integration Behavior*. Details of this structured activity are further discussed in Section 4.4.

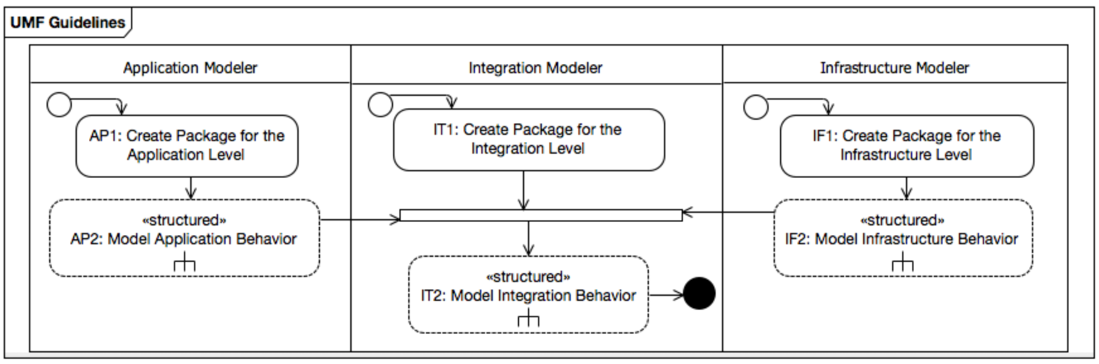
UMF Guidelines
Application Modeler
Integration Modeler
Infrastructure Modeler
AP1: Create Package for the Application Level
«structured» AP2: Model Application Behavior
IT1: Create Package for the Integration Level
«structured» IT2: Model Integration Behavior
IF1: Create Package for the Infrastructure Level
«structured» IF2: Model Infrastructure Behavior

Figure 16. Overview of CPS UML Methodology

4.2 Application Level

The application level modeling activities include three sequential steps: creating application level class diagrams (AP2.1), creating application level state machines (AP2.2) and applying the *UUP* notations on the created class and state machines (AP2.3).

A class diagram created for the application level should capture domain concepts that are needed for specifying the API information to gain access to the data and behavior of the system. It is important to mention that such a class diagram usually needs to specify *Signal*, which is a *Classifier* for specifying communication of send requests across objects. When creating a class diagram for the application level, for each class, each of its attributes captures an observable system attribute, which may be typed by a DataType in the *UUP*'s Model Libraries (Section 3) or MARTE_Library [2]. An attribute may represent a physical observation on a device (e.g., Battery Status on an X4 device). Each operation of a class in a class diagram represents either the API of the application software or an action physically performed by an operator (e.g., switching on or off of an X4 device).

Each state in a state machine created for the application level is defined, with an OCL constraint specifying its state invariants. Such an OCL constraint is constructed, based on one or more attributes of one or more classes of an application level class diagram. Each transition in a state machine should have its trigger defined as a call event corresponding to an API or a physical action defined in the class diagrams of the application level, and have its guard condition modelled as an OCL constraint on the input parameters of the trigger of the transition.

Next, application modelers need to apply *UUP* on state machines (AP2.3) to specify uncertainties and apply the UTP profile to add testing information (e.g., indicating *TestItem*).

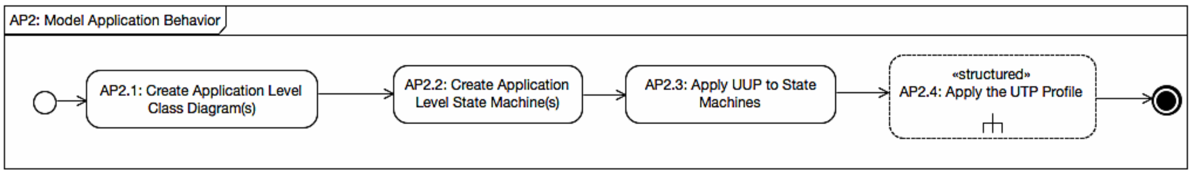
AP2: Model Application Behavior
AP2.1: Create Application Level Class Diagram(s)
AP2.2: Create Application Level State Machine(s)
AP2.3: Apply UUP to State Machines
«structured»
AP2.4: Apply the UTP Profile

Figure 17. Application Level Guideline

4.3 Infrastructure Level

For the infrastructure level, a similar modeling procedure as the one defined for the application level should be followed to derive class diagrams and state machines, apply *UUP* and the UTP profile, as shown in Figure 18. One difference is that attributes of the infrastructure level class diagrams should capture observable infrastructure attributes. For example, an attribute can be the percentage of data loss between an X4 device and the Radio Antenna. Operations of the infrastructure level class diagrams represent APIs for manipulating infrastructure level components. Regarding state machines, they should be consistent with the infrastructure level class diagrams. In other words, states should have their invariants defined as OCL constraints based on the attributes defined in the infrastructure level class diagrams, and transitions having their triggers defined as call events or time/change events.

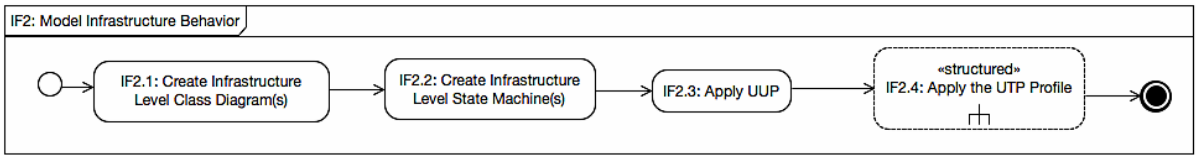
IF2: Model Infrastructure Behavior
IF2.1: Create Infrastructure Level Class Diagram(s)
IF2.2: Create Infrastructure Level State Machine(s)
IF2.3: Apply UUP
«structured»
IF2.4: Apply the UTP Profile

Figure 18. Infrastructure Level Guideline

4.4 Integration Level

Recall that, activity IT2 is started after class diagrams and state machines created for the application and infrastructure levels. As shown in Figure 19, the IT2 activity starts from creating integration level class diagrams (IT2.1) and state machines (IT2.2), followed by applying *UUP* and the UTP profile.

Regarding creating class diagrams for the integration level, such a class diagram should focus on specifying interactions between the application software and infrastructure. Particularly, signal receptions should be defined to model events that a class can receive from the infrastructure and/or application levels. Each signal reception corresponds to an instance of Signal defined in a created integration level class diagram. Notice that creating class diagrams for the integration level is not mandatory. Model elements that have been defined in the application and infrastructure level class diagrams can appear in the integration level class diagrams and they should be specified from the perspective of integration level modelers.

There are different ways of defining model elements for the integration level. One way is to refine the created application and infrastructure level state machines by directly introducing new model elements to them. For example, a state in the application level can send a Signal to the infrastructure level and vice versa. Transitions of a state machine in the application (infrastructure) level should capture triggers of type Signal Reception and effects containing Signals from the infrastructure (application) level. Another way is to keep the application and infrastructure level state machines untouched by applying a specific modeling methodology (e.g., Aspect Oriented Modeling methodologies) to specify crosscutting behaviors separately. In addition, one should also benefit from advanced features of UML State Machines (e.g., concurrent state machines, parallel regions) to for example refer to existing state machines defined in the application and infrastructure levels.

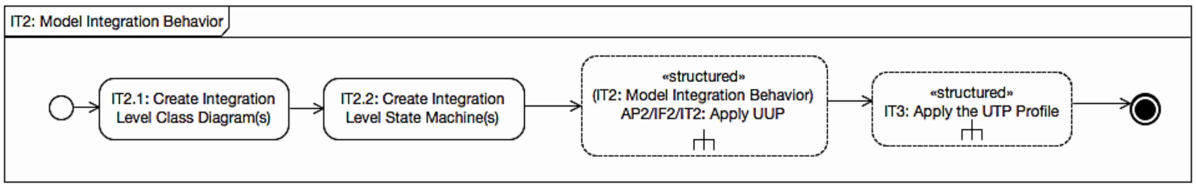
IT2: Model Integration Behavior
IT2.1: Create Integration Level Class Diagram(s)
IT2.2: Create Integration Level State Machine(s)
«structured» (IT2: Model Integration Behavior) AP2/IF2/IT2: Apply UUP
«structured» IT3: Apply the UTP Profile

Figure 19. Integration Level Guideline

4.5 Apply UML Uncertainty Profile (AP2/IF2/IT2)

Recall that the activity of applying UML Uncertainty Profile (*UUP*) is invoked at all the three levels. We tag each type of activities of the activity diagrams from Figure 20 to Figure 29 with S, C and A to represent structured activities, call behavior and normal activity nodes. As shown in Figure 20, applying *UUP* starts from applying the «BeliefElement» stereotype on any allowed state machine model element according to *UUP*. Then a modeler can optionally specify values for the "from" and "duration" attributes of the stereotype, model belief agents, model belief degree, and/or model uncertainties (Figure 20).

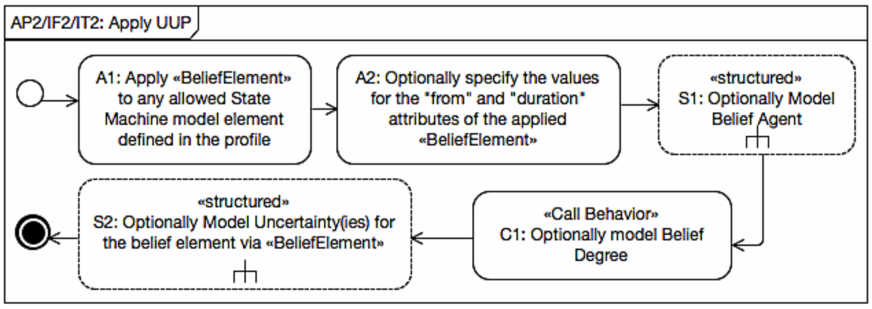
AP2/IF2/IT2: Apply UUP
A1: Apply «BeliefElement» to any allowed State Machine model element defined in the profile
A2: Optionally specify the values for the "from" and "duration" attributes of the applied «BeliefElement»
«structured» S1: Optionally Model Belief Agent
«Call Behavior» C1: Optionally model Belief Degree
«structured» S2: Optionally Model Uncertainty(ies) for the belief element via «BeliefElement»

Figure 20. Applying UUP

As shown in Figure 21, there are two ways to model belief agents (S1.1 and S1.2). A modeler can specify belief agents simply as one or more strings via the "beliefAgent" attribute of «BeliefElement» (S1.1). She/he can also create a package to organize all the belief agents (S1.2). In this case, each belief agent can be modelled as a class in the package and the package is stereotyped with «BeliefAgent». Alternatively, one can model each belief agent as a class and stereotype it with «BeliefAgent». The other option is to model each belief agent as a class and stereotype it with «BeliefAgent» and also stereotype the package with «BeliefAgent». When choosing to apply options 2, 3 and 4, one needs to link a created belief agent package to the agent attribute of «BeliefElement» (S2).

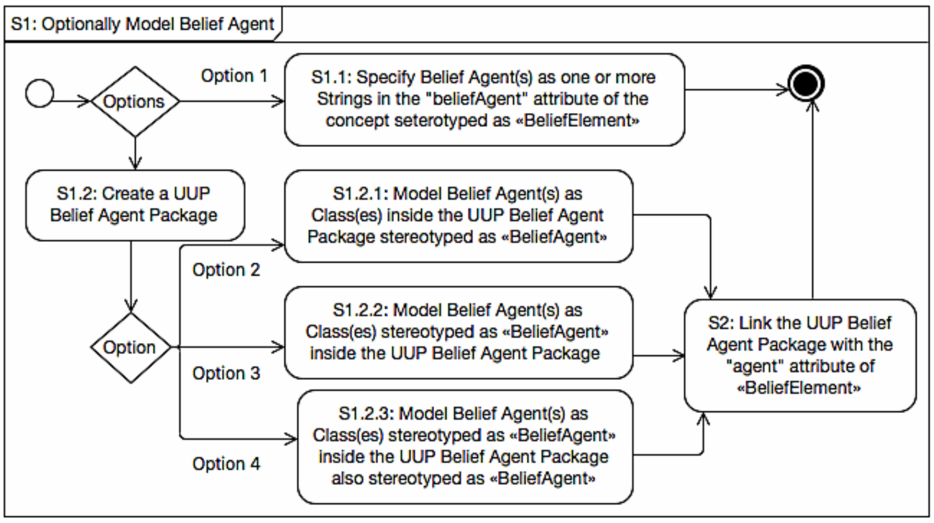
S1: Optionally Model Belief Agent
Options
Option 1
S1.1: Specify Belief Agent(s) as one or more Strings in the "beliefAgent" attribute of the concept seterotyped as «BeliefElement»
S1.2: Create a UUP Belief Agent Package
S1.2.1: Model Belief Agent(s) as Class(es) inside the UUP Belief Agent Package stereotyped as «BeliefAgent»
Option 2
Option
S1.2.2: Model Belief Agent(s) as Class(es) stereotyped as «BeliefAgent» inside the UUP Belief Agent Package
Option 3
S2: Link the UUP Belief Agent Package with the "agent" attribute of «BeliefElement»
S1.2.3: Model Belief Agent(s) as Class(es) stereotyped as «BeliefAgent» inside the UUP Belief Agent Package also stereotyped as «BeliefAgent»
Option 4

Figure 21. Model «BeliefAgent»

Modeling BeliefDegree is presented in Section 4.5.1 and modeling uncertainties is discussed in Section 4.5.2.

4.5.1 Measurement Modeling

Modeling measurements and measures are important for applying *UUP*. These activities are used to measure beliefDegree, Uncertainty, indeterminacyDegree, Risk and Effect. As shown in Figure 22, one first needs to create a package to contain measurements for indeterminacyDegree, beliefDegree, uncertaintyMeasurement, measurement of Risk and measurement of Effect (A1). Then, a modeler can optionally specify Evidence (S1), followed by the specification of each measurement instance and its corresponding measure (S3 and S2).

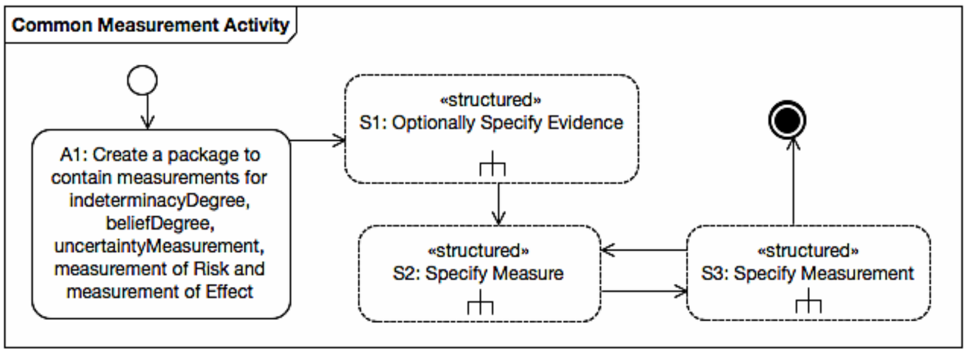
Common Measurement Activity
A1: Create a package to contain measurements for indeterminacyDegree, beliefDegree, uncertaintyMeasurement, measurement of Risk and measurement of Effect
«structured»
S1: Optionally Specify Evidence
«structured»
S2: Specify Measure
«structured»
S3: Specify Measurement

Figure 22. Common Measurement Modeling Activity

A. Specify Evidence

As shown in Figure 23, there are two ways to specify evidence. Option 1 is to specify evidence as a String value (in the “measurement” attribute of Measurement). Option 2 is to create a package for evidence if such a package does not exist and optionally stereotype it with «Evidence» (S1.2.1). One can then create any UML model element to represent evidence, according to *UUP* and optionally stereotype it with «Evidence» (S1.2.2). The last step of Option 2 is to link either the package or UML model elements representing evidence to the “referredEvidence” attribute of Measurement (S1.2.3).

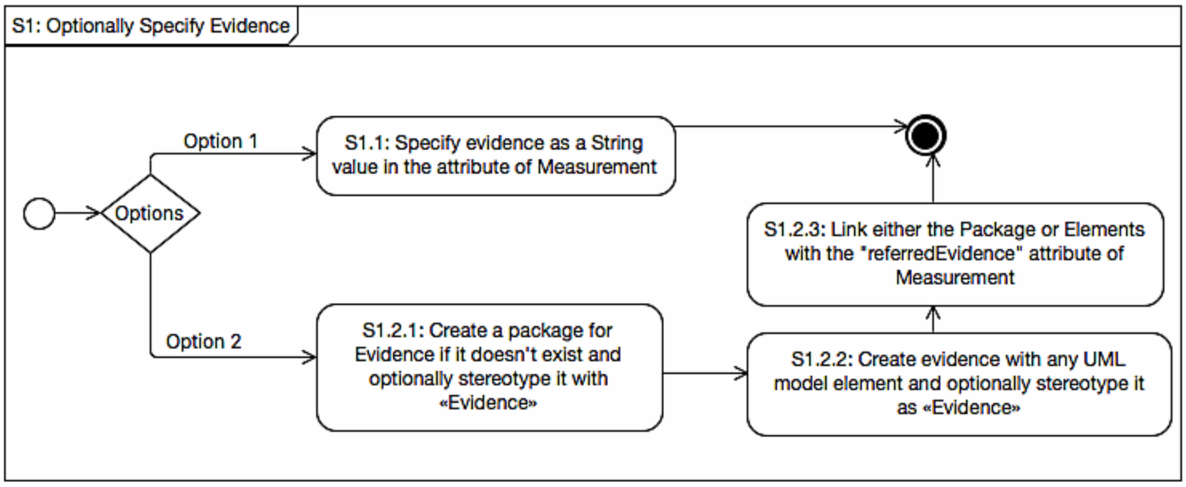
S1: Optionally Specify Evidence
Options
Option 1
Option 2
S1.1: Specify evidence as a String value in the attribute of Measurement
S1.2.1: Create a package for Evidence if it doesn't exist and optionally stereotype it with «Evidence»
S1.2.2: Create evidence with any UML model element and optionally stereotype it as «Evidence»
S1.2.3: Link either the Package or Elements with the "referredEvidence" attribute of Measurement

Figure 23. Specify Evidence

B. Specify Measure

As shown in Figure 24, to specify a measure, a modeler needs to create a class diagram (A1) and then create instances of Measures (for measurements of either "indeterminacyDegree", "beliefDegree", "uncertaintyMeasurement", measurement of Risk or measurement of Effect) as classes or datatypes (A2). One then needs to add attributes to these classes or datatypes by using the datatypes defined in the Measure Libraries. One can optionally apply corresponding measure stereotypes (e.g., «UncertaintyMeasure») to the classes or datatypes (A4). The last step is to link a measure to an instance of Measurement (A5).

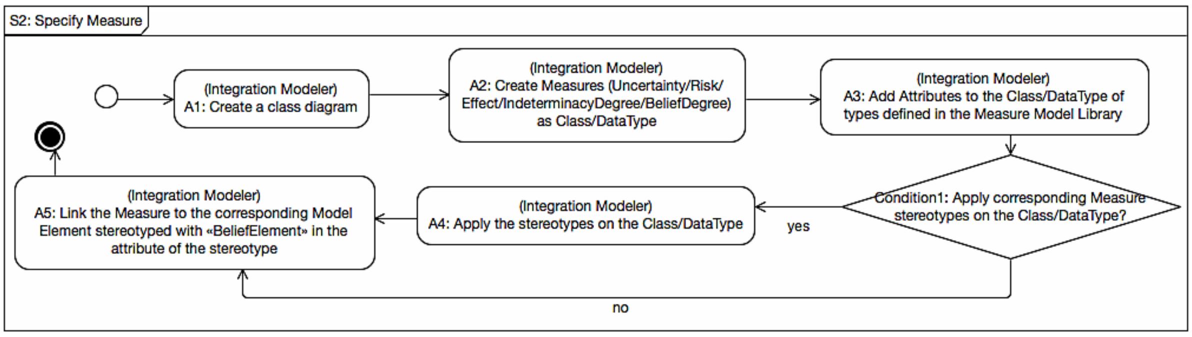
S2: Specify Measure
(Integration Modeler) A1: Create a class diagram
(Integration Modeler) A2: Create Measures (Uncertainty/Risk/ Effect/IndeterminacyDegree/BeliefDegree) as Class/DataType
(Integration Modeler) A3: Add Attributes to the Class/DataType of types defined in the Measure Model Library
Condition1: Apply corresponding Measure stereotypes on the Class/DataType?
yes
no
(Integration Modeler) A4: Apply the stereotypes on the Class/DataType
(Integration Modeler) A5: Link the Measure to the corresponding Model Element stereotyped with «BeliefElement» in the attribute of the stereotype

Figure 24. Specify Measure

C. Specify Measurement

There are three ways to specify measurements, as shown in Figure 25: specifying a measurement as a String of the measurement attribute of Measurement (A1), ValueSpecification (A2), and an OCL constraint owned by a class or datatype representing a measure, based on the attributes defined in the class or datatype (A3.1). One can also optionally apply «MeasurementConstraint» to an OCL constraint defined to specify a measurement (A3.2).

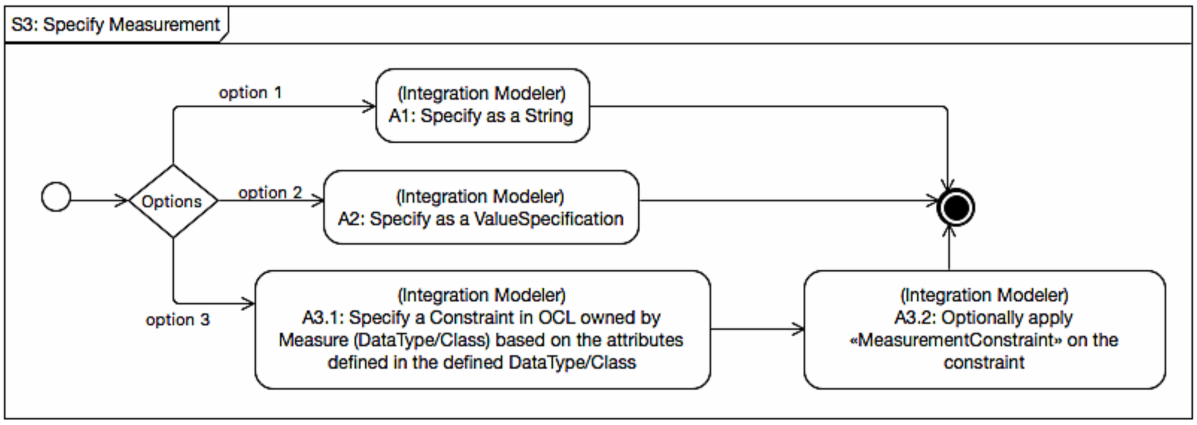
S3: Specify Measurement
Options
option 1
option 2
option 3
(Integration Modeler) A1: Specify as a String
(Integration Modeler) A2: Specify as a ValueSpecification
(Integration Modeler) A3.1: Specify a Constraint in OCL owned by Measure (DataType/Class) based on the attributes defined in the defined DataType/Class
(Integration Modeler) A3.2: Optionally apply «MeasurementConstraint» on the constraint

Figure 25. Specify Measurement

4.5.2 Uncertainty Modeling

As shown in Figure 26, one first needs to specify the kind of an uncertainty (A1), optionally specify values for attributes "from", "field", and "locality" of the uncertainty, optionally model Lifetime (or Cause, Pattern, Effect) of the uncertainty, optionally define IndeterminacySource(s), optionally model uncertaintyMeasurement and Risk.

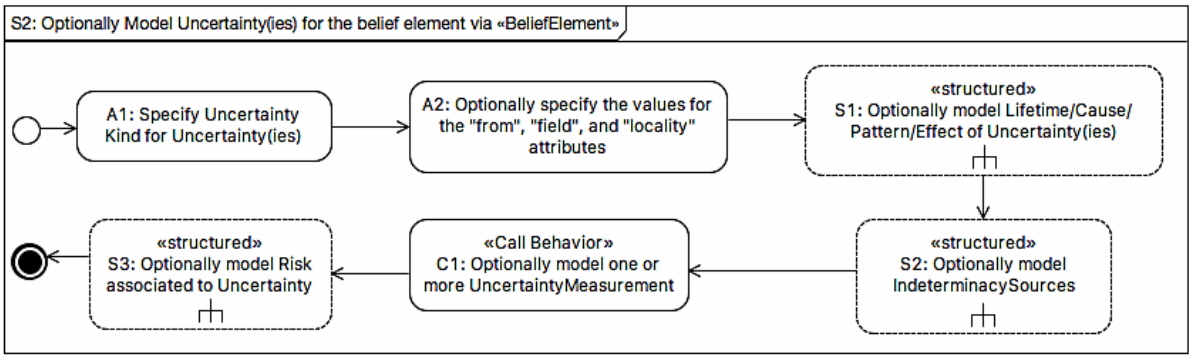
S2: Optionally Model Uncertainty(ies) for the belief element via «BeliefElement»
A1: Specify Uncertainty Kind for Uncertainty(ies)
A2: Optionally specify the values for the "from", "field", and "locality" attributes
«structured» S1: Optionally model Lifetime/Cause/ Pattern/Effect of Uncertainty(ies)
«structured» S2: Optionally model IndeterminacySources
«Call Behavior» C1: Optionally model one or more UncertaintyMeasurement
«structured» S3: Optionally model Risk associated to Uncertainty

Figure 26. Model Uncertainty

A. Model Lifetime/Cause/Pattern/Effect of Uncertainty

A modeler has two options to specify Lifetime/Cause/Pattern/Effect of an uncertainty, as shown in Figure 27. One option is to simply specify an instance of these as a String value owned by the uncertainty (via attributes "lifetime", "cause", "effect", "pattern" or "risk" of Uncertainty). The second option needs to start from creating a package for Lifetime/Cause/Pattern/Effect if such a package does not exist, and optionally apply «Lifetime», «Cause», «Pattern», or «Effect» (S1.2.1). After creating packages, one needs to create Lifetime/Cause/Pattern/Effect as any UML model element and optionally apply the corresponding stereotypes. Since Effect can be measured, an instance of it can be optionally associated with one or more measurements (Section 4.5.1). The last step of Option 2 is to associate each created package or element to corresponding attributes of Uncertainty, i.e., "referredPattern", "referredEffect", "referredLifetime", or "referredCause".

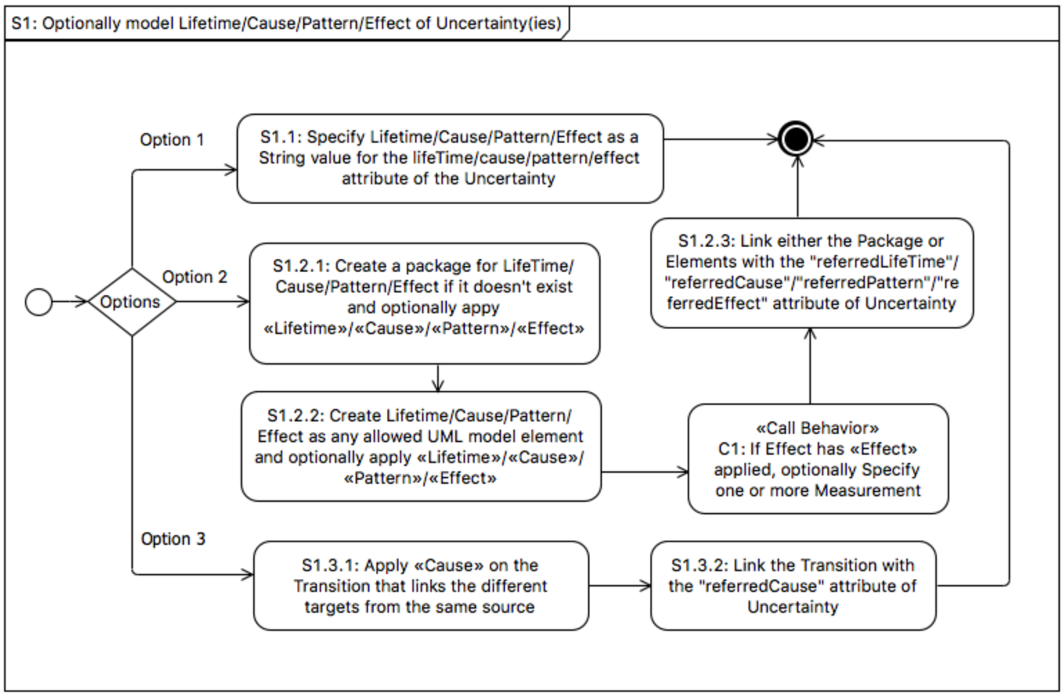
S1: Optionally model Lifetime/Cause/Pattern/Effect of Uncertainty(ies)
Option 1
Option 2
Option 3
Options
S1.1: Specify Lifetime/Cause/Pattern/Effect as a String value for the lifeTime/cause/pattern/effect attribute of the Uncertainty
S1.2.1: Create a package for LifeTime/ Cause/Pattern/Effect if it doesn't exist and optionally appy «Lifetime»/«Cause»/«Pattern»/«Effect»
S1.2.2: Create Lifetime/Cause/Pattern/ Effect as any allowed UML model element and optionally apply «Lifetime»/«Cause»/ «Pattern»/«Effect»
«Call Behavior» C1: If Effect has «Effect» applied, optionally Specify one or more Measurement
S1.2.3: Link either the Package or Elements with the "referredLifeTime"/ "referredCause"/"referredPattern"/"re ferredEffect" attribute of Uncertainty
S1.3.1: Apply «Cause» on the Transition that links the different targets from the same source
S1.3.2: Link the Transition with the "referredCause" attribute of Uncertainty

Figure 27. Model Lifetime/Cause/Patten/Effect of Uncertainty

In addition, the option 3 is newly added. Firstly, the «Cause» is mandatory to apply on the transition when it transits to the different target states (S1.3.1). The next step of option 3 is to link transition applied «Cause» to Uncertainty via "referredCause" attribute.

B. Model IndeterminacySource

As shown in Figure 28, a modeler can simply specify an indeterminacy source as a String value of attribute "indeterminacySource" of Uncertainty (Option 1). Alternatively, one can create a package to organize indeterminacy sources (A2.2.1), create instances of any UML Classifier/Constraint/Operation/Constraint to represent an indeterminacy source and apply the «IndeterminacySource» stereotype on them (A2.2.2.1-A2.2.2.4), create instances of the Constraint to represent the occurrence of the indeterminacy source and apply «IndeterminacySpecification» on them (A2.2.3). Based on the implementation of Test APIs that enable to trigger/release the occurrence of the indeterminacy source, the modeler optionally creates instances of the Operation/Behavior to represent the IndeterminacySourceInput and apply «IndeterminacySourceInput» (A2.2.5.1/ A2.2.5.2), and further link with IndeteminacySpecification by the "triggeredBy"/ "releasedBy" attribute. In addition, the modeler can specify the nature and description of each indeterminacy source (A2.2.7), specify measurements for each indeterminacy source (C1), and associate the created IndeterminacySource/IndeterminacySpecification to the "referredIndeterminacySource"/ "relatedIndSpecs" attribute of Uncertainty (A2.2.8).

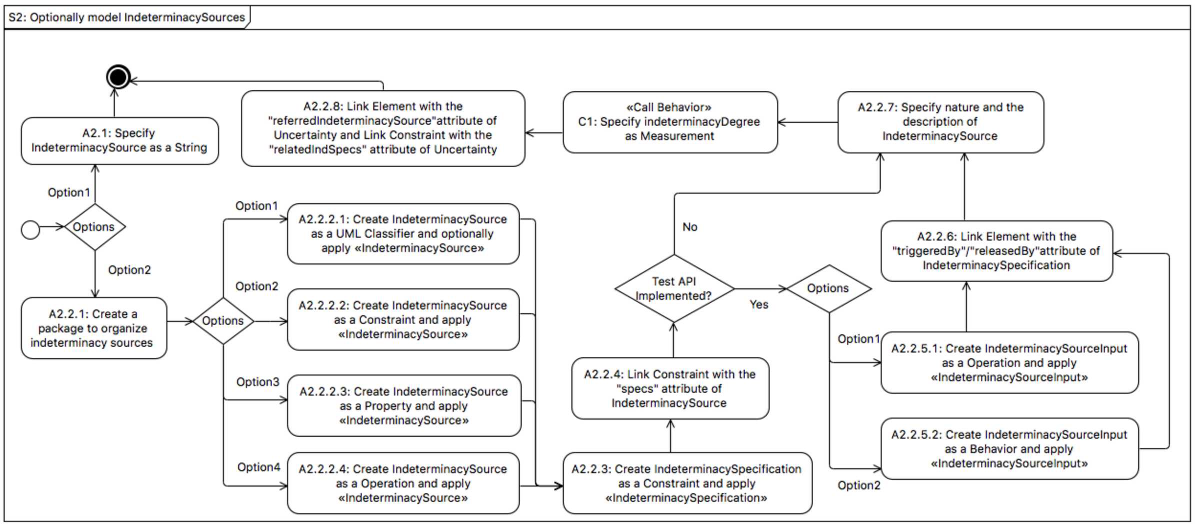
S2: Optionally model IndeterminacySources
A2.1: Specify IndeterminacySource as a String
A2.2.8: Link Element with the "referredIndeterminacySource"attribute of Uncertainty and Link Constraint with the "relatedIndSpecs" attribute of Uncertainty
«Call Behavior» C1: Specify indeterminacyDegree as Measurement
A2.2.7: Specify nature and the description of IndeterminacySource
Option1
Options
Option2
A2.2.1: Create a package to organize indeterminacy sources
Options
Option1
A2.2.2.1: Create IndeterminacySource as a UML Classifier and optionally apply «IndeterminacySource»
Option2
A2.2.2.2: Create IndeterminacySource as a Constraint and apply «IndeterminacySource»
Option3
A2.2.2.3: Create IndeterminacySource as a Property and apply «IndeterminacySource»
Option4
A2.2.2.4: Create IndeterminacySource as a Operation and apply «IndeterminacySource»
A2.2.3: Create IndeterminacySpecification as a Constraint and apply «IndeterminacySpecification»
A2.2.4: Link Constraint with the "specs" attribute of IndeterminacySource
Test API Implemented?
No
Yes
Options
Option1
Option2
A2.2.5.1: Create IndeterminacySourceInput as a Operation and apply «IndeterminacySourceInput»
A2.2.5.2: Create IndeterminacySourceInput as a Behavior and apply «IndeterminacySourceInput»
A2.2.6: Link Element with the "triggeredBy"/"releasedBy"attribute of IndeterminacySpecification

Figure 28. Model IndeterminacySource

It is possible to model these indeterminacy related concepts in different ways. Therefore, to ease the modeling process, we summarize our recommendations for applying this part of the profile in Table 4, based on our experience.

Table 4. Recommendations For applying the Indeterminacy Source part of the UUP profile

#	Stereotype Applied		Base Element
S1: States of the environment of the CPS are indeterminate, such as the batteryStatus.			
R1	«IndeterminacySource»		Property
	«IndeterminacySpecification»		Constraint
	Op1	«IndeterminacySourceInput»	Operation
	Op2	«IndeterminacySourceInput»	Operation, Constraint
R2	«IndeterminacySource»		Constraint
	«IndeterminacySpecification»		FALSE (default)
	Op1	«IndeterminacySourceInput»	Operation
	Op2	«IndeterminacySourceInput»	Operation, Constraint
S2: Input data is indeterminate.			
R1	«IndeterminacySource»		Operation
	«IndeterminacySpecification»		Constraint
	«IndeterminacySourceInput»		Constraint
S3: Occurrences of an event from the environment (e.g., "pressing the button") are indeterminate.			
R1	«IndeterminacySource»		Property
	«IndeterminacySpecification»		Constraint
	Op1	«IndeterminacySourceInput»	Operation
	Op2	«IndeterminacySourceInput»	Operation, Constraint
R2	«IndeterminacySource»		Constraint
	«IndeterminacySpecification»		FALSE (default)
	Op1	«IndeterminacySourceInput»	Operation
	Op2	«IndeterminacySourceInput»	Operation, Constraint

C. Model Risk

A modeler can also optionally associate an uncertainty to Risk. As shown in Figure 29, one can simply specify Risk as a String value of the "riskLevel" attribute of Uncertainty (Option 1) or one of the predefined risk levels in enumeration RiskLevel (Option 2). Alternatively, one can create a package for Risk if such a package does not exist, followed by creating classes and/or datatypes to represent Risks and optionally applying «Risk» (A4.3.2). Afterwards, a modeler can also optionally specify measurement for Risk (C1), and link the created classes and datatypes to Uncertainty via the "riskInDTViaClass" and/or "riskInDT" attributes (A4.3.3).

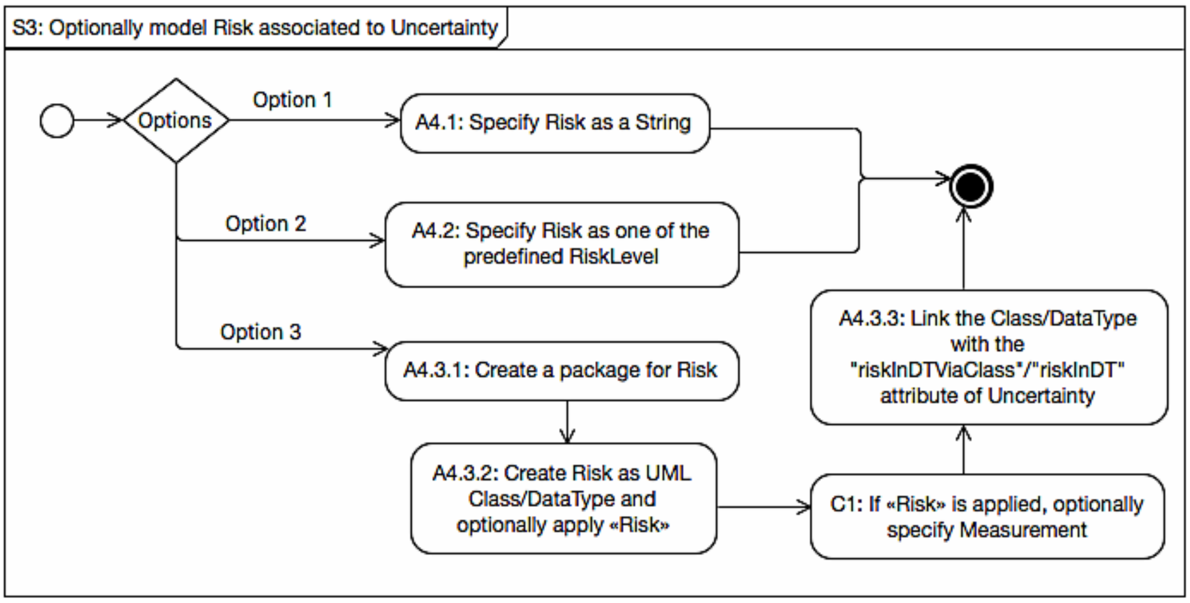
S3: Optionally model Risk associated to Uncertainty
Options
Option 1
A4.1: Specify Risk as a String
Option 2
A4.2: Specify Risk as one of the predefined RiskLevel
Option 3
A4.3.1: Create a package for Risk
A4.3.2: Create Risk as UML Class/DataType and optionally apply «Risk»
C1: If «Risk» is applied, optionally specify Measurement
A4.3.3: Link the Class/DataType with the "riskInDTViaClass"/"riskInDT" attribute of Uncertainty

Figure 29. Model Risk

ACKNOWLEDGMENT

This research was supported by the EU Horizon 2020 funded project U-Test (Testing Cyber-Physical Systems under Uncertainty). Tao Yue and Shaukat Ali are also supported by RCN funded Zen-Configurator project, RFF Hovedstaden funded MBE-CR project, RCN funded MBT4CPS project, and RCN funded Certus SFI.

REFERENCE

[1] M. Zhang, B. Selic, S. Ali, T. Yue, O. Okariz, and R. Norgren, "Understanding Uncertainty in Cyber-Physical Systems: A Conceptual Model," in ECMFA, 2016.

[2] O. M. Group, "UML Profile For MARTE: Modeling And Analysis Of Real-Time Embeded Systems," June, 2011.

[3] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *The annals of mathematical statistics*, pp. 325-339, 1967.

[4] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy sets and systems,* vol. 1, no. 1, pp. 3-28, 1978.

[5] P. Smets, and R. Kennes, "The transferable belief model," *Artificial intelligence,* vol. 66, no. 2, pp. 191-234, 1994.

[6] R. V. L. Hartley, "Transmission of information," *Bell System Technical Journal*, pp. 535-563, 1928.

[7] M. Higashi, and G. J. Klir, "Measures of uncertainty and information based on possibility distributions," *International Journal of General Systems,* vol. 9, no. 1, pp. 43-58, 1982.

[8] M. T. Lamata, and S. Moral, "Measures of entropy in the theory of evidence," *International Journal Of General System,* vol. 14, no. 4, pp. 297-305, 1988.

[9] G. Shafer, *A mathematical theory of evidence*: Princeton university press Princeton, 1976.

[10] R. R. Yager, "Entropy and specificity in a mathematical theory of evidence," *International Journal of General System,* vol. 9, no. 4, pp. 249-260, 1983.

[11] U. Höhle, "Fuzzy plausibility measures." pp. 7-30.

[12] U. Höhle, "Entropy with respect to plausibility measures." pp. 167-169.

[13] M. Negnevitsky, *Artificial intelligence: a guide to intelligent systems*: Pearson Education, 2005.

[14] L. A. Zadeh, “Fuzzy sets,” *Information and control,* vol. 8, no. 3, pp. 338-353, 1965.

[15] K. George J, and Y. Bo, “Fuzzy sets and fuzzy logic, theory and applications,” -, 2008.

[16] B. Kosko, “Fuzzy entropy and conditioning,” *Information sciences,* vol. 40, no. 2, pp. 165-174, 1986.

[17] D. Didier, and P. Henri, “Fuzzy sets and systems: Theory and Applcation.,” *Mathematics in Scince and Engineering,* vol. 144, 1980.

[18] H.-J. Zimmermann, *Fuzzy set theory—and its applications*: Springer Science & Business Media, 2011.

[19] Z. Pawlak, “Rough sets,” *International Journal of Computer & Information Sciences,* vol. 11, no. 5, pp. 341-356, 1982.

[20] J. A. Goguen, “L-fuzzy sets,” *Journal of mathematical analysis and applications,* vol. 18, no. 1, pp. 145-174, 1967.

[21] K. Atanassov, and C. Georgiev, “Intuitionistic fuzzy prolog,” *Fuzzy Sets and Systems,* vol. 53, no. 2, pp. 121-128, 1993.

[22] L. A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning—I,” *Information sciences,* vol. 8, no. 3, pp. 199-249, 1975.

[23] I. Grattan‐Guinness, “Fuzzy Membership Mapped onto Intervals and Many‐Valued Quantities,” *Mathematical Logic Quarterly,* vol. 22, no. 1, pp. 149-160, 1976.

[24] K. U. Jahn, “Intervall‐wertige Mengen,” *Mathematische Nachrichten,* vol. 68, no. 1, pp. 115-132, 1975.

[25] W. L. Gau, and D. J. Buehrer, “Vague sets,” *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 23, no. 2, pp. 610-614, 1993.

[26] A. De Luca, and S. Termini, “A definition of a nonprobabilistic entropy in the setting of fuzzy sets theory,” *Information and control,* vol. 20, no. 4, pp. 301-312, 1972.

[27] P. R. Garvey, and Z. F. Lansdowne, “Risk matrix: an approach for identifying, assessing, and ranking program risks,” *Air Force Journal of Logistics,* vol. 22, no. 1, pp. 18-21, 1998.

[28] H. Ni, A. Chen, and N. Chen, “Some extensions on risk matrix approach,” *Safety Science,* vol. 48, no. 10, pp. 1269-1278, 2010.

[29] A. S. Markowski, and M. S. Mannan, “Fuzzy risk matrix,” *Journal of hazardous materials,* vol. 159, no. 1, pp. 152-157, 2008.