The Distributions Package for SBML Level 3

Authors

Stuart L Moodie
moodie@ebi.ac.uk
EMBL-EBI
Hinxton, UK

Contributors

Nicolas Le Novère
lenov@babraham.ac.uk
Babraham Institute
Babraham, UK

Darren Wilkinson
darren.wilkinson@ncl.ac.uk
University of Newcastle
Newcastle, UK

Maciej Swat
mjswat@ebi.ac.uk
EMBL-EBI
Hinxton, UK

Sarah Keating
skeating@ebi.ac.uk
EMBL-EBI
Hinxton, UK

Colin Gillespie
c.gillespie@ncl.ac.uk
University of Newcastle
Newcastle, UK

10 January, 2013

Version 0.10 (Draft)

Disclaimer: This is a working draft of the SBML Level 3 "distib" package proposal. It is not a normative document. Please send comments and other feedback to the mailing list: sbml-distrib@lists.sourceforge.net.

Contents

Revision History

Version	Date	Author	Comments
0.1 (Draft)	15 Oct 2011	Stuart Moodie	First draft
0.2 (Draft)	16 Oct 2011	Stuart Moodie	Added introductory text and background info. Other minor changes etc.
0.3 (Draft)	16 Oct 2011	Stuart Moodie	Filled empty invocation semantics section.
0.4 (Draft)	4 Jan 2012	Stuart Moodie	Incorporated comments from NlN, MS and SK. Some minor revisions and corrections.
0.5 (Draft)	6 Jan 2012	Stuart Moodie	Incorporated addition comments on aim of package from NlN.
0.6 (Draft)	19 Jul 2012	Stuart Moodie	Incorporated revisions discussed and agreed at HARMONY 2012.
0.7 (Draft)	6 Aug 2012	Stuart Moodie	Incorporated review comments from Maciej Swat and Sarah Keating.
0.8 (Draft)	21 Dec 2012	Stuart Moodie	Incorporated changes suggested at combine and subsequently through list discussions.
0.9 (Draft)	9 Jan 2012	Stuart Moodie	Incorporated corrections and comments from Maciej Swat and Sarah Keating.
0.10 (Draft)	10 jan 2012	Stuart Moodie	Modified based on comments from MS.

1 Introduction and motivation

1.1 What is it?

The Distributions package (also affectionately known as distrib for short) provides an extension to SBML Level 3 that enables it to encode defines both discrete and continuous probability distributions. The the distributions are only used to sample a random value from. Applications of the package include for instance descriptions of population based models: an important subset of which are pharmacokinetic/pharmacodynamic (PK/PD) models[1], which are used to model the action of drugs.

Note that originally the package was called Distributions and Ranges, but Ranges and the use of probability distributions to describe statistical uncertainly are no longer in the scope: this hence the name change.

1.2 Scope

The Distributions package adds support to SBML for sampling from a probability distribution. In particular the following are in scope:

- Sampling from a continuous distribution.
- Sampling from a discrete distribution.
- Sampling from user-defined probability density function.
- Sampling from user-defined discrete probability density function.

At one point the following were considered for inclusion in this package but are now **out of scope**:

- Stochastic differential equations.
- Other functions used to characterise a probability distribution, such as cumulative distribution functions (CDF) or survival functions etc.
- The specification of descriptive statistics (mean, standard deviation, standard error etc).[2]

1.3 This Document

This proposal describes the consensus view of workshop participants and subscribers to the sbml-distrib mailing list. Although it was written by the listed author(s) it does not reflect their views not is it their proposal. Rather, it is their understanding of the consensus view of what the Distributions package should do and how it should do it. The contributors listed have made significant contributions to the development and writing of this specification and are credited accordingly, but a more comprehensive attribution is provided in the acknowledgements (section 8).

Finally, there are issues that have arisen during the writing of this document (section 7). It is important that they are considered and resolved and so the author(s) would encourage the reader to consider them and contribute their ideas or comments — indeed any feedback about this proposal — to the distrib discussion list[3].

Once the proposal is finalised this will be the first step towards the formal adoption of the distrib as a package in SBML Level 3. After this two implementations based on this proposal are required and then a vote on its adoption by the SBML community. The proposal will then provide the basis for a future package specification document. More details of the SBML package adoption process can be found at: http://sbml.org/Documents/SBML_Development_Process.

[1] for more information see: http://www.pharmpk.com/.
[2] It is proposed that this be provided descriptive statistics in a separate package.
[3] sbml-distrib@lists.sourceforge.net

Please also note that in this draft of the proposal a list of the specific probability distributions to be supported has not been included. This is because at present it is envisaged that distrib will refer to an external definition of probability distributions (see sections 3.10 and 7).

1.4 Conventions used in this document

As we are early in the package proposal process there will be some parts of this proposal where there is no clear consensus on the correct solution or only recent agreement or agreement by a group which may not be representative of the SBML community as a whole. These cases are indicated by the question mark in the left margin (illustrated). The reader should pay particular attention to these points and ideally provide feedback, especially if they disagree with what is proposed. Similarly there will be points — especially as the proposal is consolidated — which are agreed, but which the reader should take note of and perhaps read again. These points are emphasised by the hand pointer in the left margin (illustrated).

2 Background

2.1 Problems with current SBML approaches

SBML Level 3 Core has no direct support for encoding random values within a model. Currently there is no workaround within the core language itself, although it is possible to define such information using annotations within SBML itself. Frank Bergmann had proposed such an semi-formalised extension for use with SBML L2 [REF?].

2.2 Past work on this problem or similar topics

2.2.1 The Newcastle Proposal

In 2005 there was a proposal from Colin Gillespie and others [4] to introduce support for probability distributions in the SBML core specification. This was based on their need to use such distributions to represent the models they were creating as part of the BASIS project http://www.basis.ncl.ac.uk).

They proposed that distributions could be referred to in SBML using the **csymbol** element in the MathML subset used by the SBML Core specification. An example is below:

```
<xmlns=''http://www.w3.org/1998/Math/MathML''>
  <apply>
    <csymbol encoding=''text''
        definitionURL=''http://www.sbml.org/sbml/symbols/uniformRandom''>
      uniformRandom
    </csymbol>
    <ci>mu</ci>
    <ci>sigma</ci>
  </apply>
</math>
```

This required that a library of definitions be maintained as part of the SBML standard and in their proposal they defined an initial small set of commonly used distributions. The proposal was never implemented.

2.2.2 Seattle 2010

The "distrib" package was discussed at the Seattle SBML Hackathon[5] and this section is an almost verbatim reproduction of Darren Wilkinson's report on the meeting[6]. There Darren presented an overview of the problem[7][8], building on the old proposal from the Newcastle group (see above: 2.2.1). There was broad support at the meeting for development of such a package, and for the proposed feature set. Discussion following the presentation led to a consensus on the following points:

- There is an urgent need for such a package.
- It is important to make a distinction between a description of uncertainty regarding a model parameter and the mechanistic process of selecting a random number from a probability distribution, for applications such as parameter scans and experimental design
- It is probably worth including the definition of PMFs, PDFs and CDFs in the package
- It is worth including the definition of random distributions using particle representations within such a package, though some work still needs to be done on the precise representation

[4] http://sbml.org/Community/Wiki/SBML_Leve\T1\l_3_Proposals/Distributions_and_Ranges
[5] http://sbml.org/Events/Hackathons/The_2010_SBML-BioModels.net_Hackathon
[6] http://sbml.org/Forums/index.php?t=tree&goto=6141&rid=0
[7] Slides: http://sbml.org/images/3/3b/Djw-sbml-hackathon-2010-05-04.pdf
[8] Audio: http://sbml.org/images/6/67/Wilkinson-distributions-2010-05-04.mov

- It could be worth exploring the use of xinclude to point at particle representations held in a separate file
- Random numbers must not be used in rate laws or anywhere else that is continuously evaluated, as then simulation behaviour is not defined
- Although there is a need for a package for describing extrinsic noise via stochastic differential equations in SBML, such mechanisms should not be included in this package due to the considerable implications for simulator developers
- We probably don't want to layer on top of UncertML (www.uncertml.org), as this spec is fairly heavy-weight, and somewhat tangential to our requirements
- A random number seed is not part of a model and should not be included in the package
- The definition of truncated distributions and the specification of hard upper and lower bounds on random quantities should be considered.

It was suggested that new constructs should be introduced into SBML by the package embedded as user-defined functions using the following syntax:

```
<listOfFunctionDefinitions>
  <functionDefinition id="myNormRand">
    <distrib:####>
      #### distrib binding information here ####
    </distrib:####>
    <math>
      <lambda>
        <bvar>
          <ci>mu</ci>
          <ci>sigma</ci>
        </bvar>
        <ci>mu</ci>
      </lambda>
    </math>
  </functionDefinition>
</listOfFunctionDefinitions>
```

which allows the use of a "default value" by simulators which do not understand the package (but simulators which do will ignore the <math> element). The package would nevertheless be "required", as it will not be simulated correctly by software which does not understand the package.

Informal discussions following the break-out covered topics such as:

- how to work with vector random quantities in the absence of the vector element in the MathML subset used by SBML
- how care must be taken with the semantics of random variables and the need to both:
  - reference multiple independent random quantities at a given time
  - make multiple references to the same random quantity at a given time.

2.2.3 Hinxton 2011

Detailed discussion was continued at the Statistical Models Workshop in Hinxton in June 2011[9]. There those interested in representing Statistical Models in SBML came together to work out the details of how this package would work in detail. Dan Cornford from the UncertML project[10] attended the meeting and described how that resource could be used to describe uncertainty and in particular probability distributions. Perhaps the most significant

[9] http://sbml.org/Events/Other_Events/statistical_models_workshop_2011
[10] http://www.uncertml.org/

decision at this meeting was to adopt the UncertML resource as a controlled vocabulary that is referenced by the Distributions package.

Much has changed since this meeting, but the output from this meeting was the basis for the first version of this proposal.

2.2.4 HARMONY 2012: Maastricht

Two sessions were dedicated to discussion of Distributions at HARMONY based around the proposals described in version 0.5 of this document. In addition there was discussion about the Arrays and Sets proposal which was very helpful in solving the problem of multivariate distributions in Distributions. The following were the agreed outcomes of the meeting:

- The original proposal included UncertML markup directly in the function definition. This proved unwieldy and confusing and has been replaced by a more elegant solution that eliminates the UncertML markup and integrates well with the fallback function (see details below).
- Multivariate distributions can be supported using the Arrays and Sets package to define a covariance matrix.
- User defined continuous distributions would define a PDF in MathML.
- Usage semantics were clarified so that invokation of a function definition implied a value was sampled from the specified distribution.
- It was agreed from which sections of an SBML model a distribution could be invoked.
- Statistical descriptors of variables (for example mean and standard deviation) would be separated from Distributions and either provided in a new package or in a later version of SBML L3 core.

2.2.5 COMBINE 2012: Toronto

The August proposal was reviewed and an improvement was agreed to the user-defined PMF part of the proposal. In particular is was agreed that the categories should be defined by distrib classes rather than by passing in the information as an array. Questions were also raised about whether UncertML was suitably well defined to be used as an external definition for probability distributions. This was resolved subsequent to the meeting with a teleconference to Dan Cornford and colleagues. These changes are incorporated here. Finally, there was considerable debate about whether to keep the dependence of distrib on the Arrays package in order to support multi-variate distributions. The outcome was an agreement that we would review this at the end of 2012, based on the results of an investigation into how feasible it would be to implement Arrays and Sets as a package.

3 Proposed syntax and semantics

3.1 Overview

This section describes the new elements that are provided by the class to SBML and existing SBML elements that are modified in some way by the package. Although distrib, like SBML is implemented in XML, we follow the convention of the SBML specification and describe the package in terms of UML classes and attributes.

Throughout this document, in all UML diagrams, classes that exists in SBML Level 3 Version 1 or another existing standard are displayed in black. If those elements are extended in this proposal, those extensions are displayed in green. Classes that are new to this proposal are shown in blue.

3.2 Defining Distributions

3.2.1 The approach

The Distributions package has a very simple purpose. It provides a mechanism for sampling a random value from a probability distribution. This implies that it must achieve two tasks. It must first define the probability distribution and next it must sample a random value from the distribution.

There are three ways to define a probability distribution in distrib. The first is to reuse an existing definition. This definition is external to the distrib specification, and in theory any definition that can be specified by a URI is permitted. However, to promote inter-operability it is recommended that UncertML is used as the source of external definitions and this proposal document assumes its use.

If not using a pre-defined definition then one must explicitly define the probability distribution, either as a probability density function (PDF) for a continuous distribution or a probability mass function (PMF) for a discrete one. The Distributions package provides a mechanism to do this by referencing an external definition.

Strictly all we need are the explicit PDF and PMF definitions. However, the advantage of using a pre-defined distribution is that software can easily recognise the distribution and use an optimised built-in implementation rather than interpreting the distribution from the PDF and PMF definitions. For some applications such optimisations make important performance differences.

As mentioned above the distribution must be sampled. In distrib the distributions are sampled when they are invoked. To reuse a sampled value the value must be assigned to a paramater first. The implementation algorithm used is not specified. The package permits the definition of truncated distributions when using externally defined distributions and again how this is implemented is left to the implementor.

3.2.2 Design Overview

SBML Core is extended by redefining the **FunctionDefinition** class as can be seen in the UML representation in figure 1 on the following page. The redefined **FunctionDefinition** can optionally contain a single instance of either **externalDefinition**, **explicitPDF** or **explicitPMF**. These classes describe the externally defined probability distribution, the explicitly defined PDF or the explicitly defined PMF respectively.

Slightly anomalously the **FunctionDefinition** class also contains a MathML block containing a standard SBML function definition. This is required to comply with the *Validity after Reduction* rule in the package design guidelines SBML Editorial Board (2012) and ensures a degree of backwards compatibility for SBML readers and validators that do not understand the distrib package.

3.3 Package Usage

The Distributions package is defined as a **required** package. This means that unless the software reading the SBML document 'understands' distrib the model cannot be guaranteed to be correct. The package is targetting SBML Level 3 Version 1 and so complies with the package design guidelines for that version SBML Editorial Board (2012).

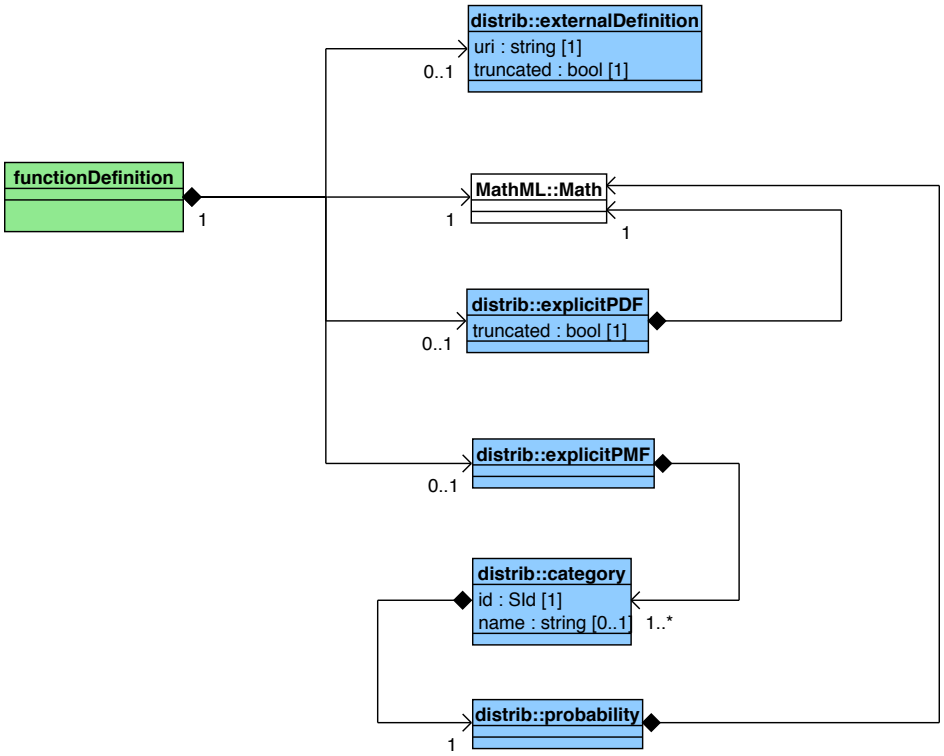

Figure 1: *UML class diagram for the Distributions package. This diagram describes the classes involved and not instances of those classes. The namespace of the SBML Core is not shown.*

3.4 FunctionDefinition Extension

As outlined above the **FunctionDefinition** is redefined to contain the following classes from distrib. It maintains it existing associations and so must also contain a MathML block defining a lambda function. In normal usage it is ignored, but it has the potential to be invoked by software that does not support distrib so it should be semantically correct (see section 3.11).

- **externalDefinition**
- **explicitPDF**
- **explicitPMF**

3.5 externalDefinition

Defining the external definition requires the specification of a URI that defines a probability distribution. A truncated version of the distribution may be used, in which case the `truncated` attribute is true. An overview of the class specification is shown in figure 2 and the attributes of the class are defined below.

It was discussed above that the recommended source of probability distribution definitions is UncertML. In some cases UncertML permits some flexibility in its description of some distributions and permits the invocation of several probability distributions in one definition. To avoid this and to provide clarity and appendix (section A) is provided to this document describing the available distributions and how they can be used by distrib. It is recommended that only those distribution in the appendix are used.

`uri`

This provides the external definition of the distribution to be invoked. The value of the attribute should be a URI that uniquely identifies a definition from an external resource. The exact nature of this URI and the external

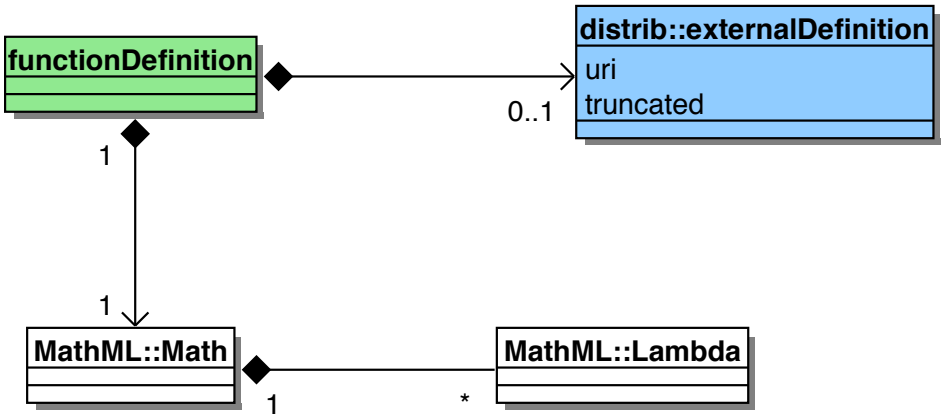

Figure 2: *UML class diagram describing the* ***externalDefinition*** *class and its association with* ***functionDefinition****.*

resource to use is discussed below (section 3.5). The attribute is mandatory.

`truncated`

This attribute indicates that a distribution is to be truncated, if true or not if false. How this truncation is to be implemented is tool specific. When set to true an additional two arguments are implied when invoking the distribution. These arguments are appended to those already required by the distribution and are invoked in the order they are described below. In cases where one only wants to define and upper or lower bound to the distribution values of – inf and inf should be used respectively.

lower limit The lower boundary of the truncation.

upper limit The upper boundary of the truncation.

3.6 explicitPDF

This class describes a continuous probability distribution by explicitly defining its probability density function (PDF). The PDF is described mathematically using a MathML definition contained by the **explicitPDF** class. Note that this function definition is distinct from the fallback function described in section 3.11.

3.7 explicitPMF

A discrete probability distribution is described by **explicitPMF** and its associated classes, **category** and **probability**. Together they define a probability mass function (PMF) where each category can have an associated probability value (figure 3).

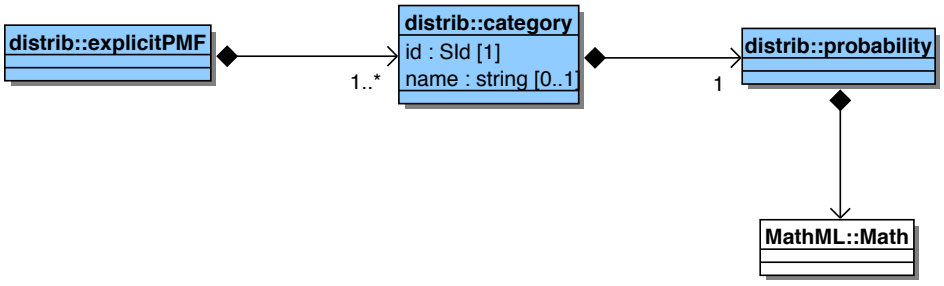

Figure 3: *UML class diagram describing the* ***explicitPMF*** *class and it's associated classes used to describe each category.*

3.7.1 category

The **category** class is used to define a discrete category in the PMF. It is identified by an `id`, which should be unique within the PMF definition and a `name` that is intended to be human readable and meaningful. It has the following attributes.

`id`

An identifier for this category of type `SId`. Note that this id is scoped by the **explicitPMF**. In other-words the id must be unique within the set of categories contained by an instance of **explicitPMF**.

`name`

The name of the category that is intended to be human readable and meaningful. It is of type `string`. The name is optional and does not need to be unique.

3.7.2 probability

The **probability** class holds the mathematical definition of the probability value associated with a category. This value is defined by an associated MathML definition. Using MathML to define the probability gives the maximum flexibility as it permits the value to be obtained by the evaluation of a mathematical expression, as a numerical constant or by referencing a variable, which could be a parameter for example.

3.8 Using the distrib package

The distribution defined by a **FunctionDefinition** instance can be used by instances of the following SBML classes:

- InitialAssignment
- EventAssignments
- Delays
- Priorities

An invocation of a distribution behaves like any other function call in SMBL and the function is executed. For a statistical distribution this means that one or more random values are sampled from the distribution each time the function definition is invoked. Each invocation implies one sampling operation. Note thatincluding sampling in the event related classes (i.e.; EventAssignments, Delays and Priorities) may be problematic in some circumstances and may unexpected and or unstable behaviour of the model. This has been raised in the issues section (7).

3.9 Permitted Types

The Distributions package will support arrays and matrices via the Arrays and Sets package. This means that paramaters passed to a function definition can be arrays, matrices or scalar values and that a functions return type can be either a scalar or the above non-scalar types. Type consistency and conversion behaviour between scalar and non-scalar types is defined by the Arrays and Sets package. See unresolved issues (section 7).

3.10 External Definition of Distributions

The external definition is a URI that should refer to a standardised dictionary of distributions. It should unambiguously define the following:

- A globally unique URI by which to refer to it.
- The parameter arguments used by the function, including for each argument: its name and type (scalar or non-scalar). Note that all arguments are mandatory.
- The type of the random value sampled from the probability distribution.

- An explicit and detailed mathematical description of the statistical distribution.

The recommended external definition for distributions is UncertML: as discussed above (section 3.5). Appendix A clarifies how the distribution definitions should be invoked and the sampled value to expect when invoked as a **functionDefinition**.

3.11 Equivalence with Fallback Function

The MathML definition directly contained by the **functionDefinition** is not used and is provided solely to satisfy the *validity upon reduction* rule for packages SBML Editorial Board (2012). This rules states that the SBML document must be syntactically valid if all package specific elements are removed from it. To ensure that this is the case the fallback function used in relation to distrib must satisfy the following rules:

- the lambda function should have the same number of arguments as its equivalent distribution (defined by distrib).
- Each argument should match the type of the equivalent argument in the external function.
- The lambda function should have the same return type as the *sampled* distribution. For example an explicit PDF when sampled will return a scalar value, in which case the dummy function should also do so.

Clearly these rules can only be enforced by a distrib aware validator.

4 Package dependencies

This package is dependent on the Arrays and Sets package to provide array and matrix structures. It is also dependent on MathML Ausbrooks et al. (2003) to define distributions explicitly. It uses the the subset of MathML set out in the SMBL Level 3 Core Specification Hucka et al. (2010).

5 Use-cases and examples

5.0.1 Sampling from a distribution: PK/PD Model

This is a very straightforward use of an externally defined distribution. The key point to note is that a value is sampled from the distribution and assigned to a variable when it is invoked. In the initialAssignments element in this example. Later use of the variable does not result in re-sampling from the distribution. This is consistent with current SBML semantics.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
    xmlns:distrib="http://www.sbml.org/sbml/level3/version1/distrib/version1"
    distrib:required="true">
    <model id="PKModel" name="PKModel" substanceUnits="item"
          timeUnits="second" volumeUnits="litre" extentUnits="item">
        <listOfCompartments>
            <compartment id="central" name="central" size="0" constant="true"/>
            <compartment id="gut" name="gut" size="0" constant="true"/>
        </listOfCompartments>
        <listOfSpecies>
            <species id="Qc" compartment="central" initialAmount="1"
               hasOnlySubstanceUnits="true" boundaryCondition="false" constant="false"/>
            <species id="Qg" compartment="gut" initialAmount="1"
               hasOnlySubstanceUnits="true" boundaryCondition="false" constant="false"/>
        </listOfSpecies>
        <listOfParameters>
            <parameter id="ka" value="1" constant="true">
    <distrib:normal tranformation="log">
      <distrib:mean transformed="false">
        <distrib:varref idref="ka"/>
      </distrib:mean>
      <distrib:stddev transformed="false">
        <distrib:real value="0.1"/>
      </distrib:stddev>
    </distrib:normal>
      </parameter>
            <parameter id="ke" value="1" constant="true"/>
            <parameter id="Cc" value="1" constant="false"/>
            <parameter id="Cc_obs" value="1" constant="false"/>
        </listOfParameters>
        <listOfInitialAssignments>
            <initialAssignment symbol="central">
                <math xmlns="http://www.w3.org/1998/Math/MathML">
                      <cn>0.5</cn>
                </math>
    <distrib:normal tranformation="log">
      <distrib:mean transformed="false">
        <distrib:real value="0.5"/>
      </distrib:mean>
      <distrib:stddev transformed="false">
        <distrib:real value="0.1"/>
      </distrib:stddev>
    </distrib:normal>
            </initialAssignment>
            <initialAssignment symbol="ka">
                <math xmlns="http://www.w3.org/1998/Math/MathML">
                    <apply>
                        <cn>0.5</cn>
                    </apply>
                </math>
    <distrib:normal tranformation="log">
      <distrib:mean transformed="false">
        <distrib:real value="0.5"/>
      </distrib:mean>
```

```
      <distrib:stddev transformed="false">
        <distrib:real value="0.1"/>
      </distrib:stddev>
    </distrib:normal>
            </initialAssignment>
            <initialAssignment symbol="ke">
                <math xmlns="http://www.w3.org/1998/Math/MathML">
                    <apply>
                        <ci>logNormal</ci>
                        <cn>0.5</cn>
                        <cn>0.1</cn>
                    </apply>
                </math>
            </initialAssignment>
        </listOfInitialAssignments>
        <listOfRules>
            <assignmentRule variable="Cc">
                <math xmlns="http://www.w3.org/1998/Math/MathML">
                    <apply>
                        <divide/>
                        <ci> Qc </ci>
                        <ci> central </ci>
                    </apply>
                </math>
            </assignmentRule>
            <assignmentRule variable="Cc_obs">
                <math xmlns="http://www.w3.org/1998/Math/MathML">
                    <apply>
                        <plus/>
                        <ci> Cc </ci>
                        <cn type="integer"> 1 </cn>
                    </apply>
                </math>
            </assignmentRule>
        </listOfRules>
        <listOfReactions>
            <reaction id="absorption" reversible="false" fast="false">
                <listOfReactants>
                    <speciesReference species="Qg" stoichiometry="1" constant="false"/>
                </listOfReactants>
                <listOfProducts>
                    <speciesReference species="Qc" stoichiometry="1" constant="false"/>
                </listOfProducts>
                <kineticLaw>
                    <math xmlns="http://www.w3.org/1998/Math/MathML">
                        <apply>
                            <times/>
                            <ci> ka </ci>
                            <ci> Qg </ci>
                        </apply>
                    </math>
                </kineticLaw>
            </reaction>
            <reaction id="excretion" reversible="false" fast="false">
                <listOfReactants>
                    <speciesReference species="Qc" stoichiometry="1" constant="false"/>
                </listOfReactants>
                <kineticLaw>
                    <math xmlns="http://www.w3.org/1998/Math/MathML">
                        <apply>
                            <divide/>
                            <apply>
                                <times/>
                                <ci> ke </ci>
                                <ci> Qc </ci>
                            </apply>
                            <ci> central </ci>
```

```
                        </apply>
                    </math>
                </kineticLaw>
            </reaction>
        </listOfReactions>
    </model>
</sbml>
```

5.1 Truncated distribution

To encode a truncated distribution we rely on external definitions. Clearly it would be cumbersome if every distribution and multivariate distribution required a truncated equivalent definition, but at present this is what is required. Perhaps this problem could be solved if optional function arguments were permitted, but this cases problems with the fallback function (see section 7).

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
      xmlns:distrib="http://www.sbml.org/sbml/level3/version1/distrib/version1"
      distrib:required="true">
  <!-- Note that this is a code snippet and not a valid model -->
  <model>
    <listOfFunctionDefinitions>
      <!-- We treat truncated distributions as any other distribution. The externally
     defined function handles the truncation. -->
      <functionDefinition id="normal">
  <distrib:externalDefinition uri="http://www.uncertml.org/distributions/normal" truncated="true"/>
  <math xmlns="http://www.w3.org/1998/Math/MathML"
        xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
    <lambda>
      <bvar><ci>mu</ci></bvar>
      <bvar><ci>sigma</ci></bvar>
      <bvar><ci>lower</ci></bvar>
      <bvar><ci>upper</ci></bvar>
      <apply>
        <ci>mu</ci>
      </apply>
    </lambda>
  </math>
      </functionDefinition>
    </listOfFunctionDefinitions>
    <listOfParameters>
      <parameter id="V" constant="true"/>
      <parameter id="V_pop" constant="true"/>
      <parameter id="V_omega" constant="true"/>
      <parameter id="V_upper" constant="true"/>
      <parameter id="V_lower" constant="true"/>
    </listOfParameters>
    <initialAssignments>
      <initialAssignment symbol="V_pop">
  <math xmlns="http://www.w3.org/1998/Math/MathML"
        xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
    <apply>
      <cn>105</cn>
    </apply>
  </math>
      </initialAssignment>
      <initialAssignment symbol="V_upper">
  <math xmlns="http://www.w3.org/1998/Math/MathML"
        xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
    <apply>
      <cn>150</cn>
    </apply>
  </math>
      </initialAssignment>
```

```
      <initialAssignment symbol="V_lower">
  <math xmlns="http://www.w3.org/1998/Math/MathML"
        xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
    <apply>
      <cn>15</cn>
    </apply>
  </math>
      </initialAssignment>
      <initialAssignment symbol="V_omega">
  <math xmlns="http://www.w3.org/1998/Math/MathML"
        xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
    <apply>
      <cn>0.70</cn>
    </apply>
  </math>
      </initialAssignment>
      <initialAssignment symbol="V">
  <math xmlns="http://www.w3.org/1998/Math/MathML"
        xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
    <apply>
      <ci>normal</ci>
      <ci>V_pop</ci>
      <ci>V_omega</ci>
      <ci>V_upper</ci>
      <ci>V_lower</ci>
    </apply>
  </math>
      </initialAssignment>
    </initialAssignments>
  </model>
</sbml>
```

5.2 Multivariate distribution

In this example two correlated parameters are sampled from a multivariate distribution. The correlation is defined using a covariance matrix and the sampled values are returned as a vector of 2 values, which are then assigned to the individual parameters. This example relies heavily on the arrays package, which will be a dependency of the distrib package. Note that because the arrays proposal is under development this example may be completely consistent with it.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
      xmlns:distrib="http://www.sbml.org/sbml/level3/version1/distrib/version1"
      distrib:required="true"
      xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1"
      arrays:required="true">
  <!-- NOTE: This requires the arrays package! -->
  <model id="MultivariateExample" name="Multivariate␣Example" substanceUnits="item"
   timeUnits="second" volumeUnits="litre" extentUnits="item">
    <listOfFunctionDefinitions>
      <functionDefinition id="multivariateNormal">
  <distrib:externalDefinition uri="http://identifiers.org/Distribution/MultivariateNormal"/>
  <math xmlns="http://www.w3.org/1998/Math/MathML"
        xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
    <lambda>
      <bvar><ci>meanVector</ci></bvar>
      <bvar><ci>covarianceMatrix</ci></bvar>
      <bvar><ci>lowerVector</ci></bvar>
      <bvar><ci>upperVector</ci></bvar>
      <apply>
        <ci>meanVector</ci>
      </apply>
    </lambda>
```

```
</math>
    </functionDefinition>
  </listOfFunctionDefinitions>
  <listOfParameters>
    <parameter id="V">
      <arrays:ArrayParameter>

      </arrays:ArrayParameter>
    </parameter>
    <parameter id="V" constant="true"/>
    <parameter id="V_pop" constant="true"/>
    <parameter id="V_omega" constant="true"/>
    <parameter id="Cl" constant="true"/>
    <parameter id="Cl_pop" constant="true"/>
    <parameter id="Cl_omega" constant="true"/>
    <parameter id="covariance" constant="true">
<arrays:listOfDimensions>
  <arrays:dimension id="i" lowerLimit="1" upperLimit="2" />
  <arrays:dimension id="j" lowerLimit="1" upperLimit="2" />
</arrays:listOfDimensions>
    </parameter>
    <parameter id="correlated_means" constant="true">
<arrays:listOfDimensions>
  <arrays:dimension id="i" lowerLimit="1" upperLimit="2" />
</arrays:listOfDimensions>
    </parameter>
    <parameter id="limit_vector" constant="true">
<arrays:listOfDimensions>
  <arrays:dimension id="i" lowerLimit="1" upperLimit="2" />
</arrays:listOfDimensions>
    </parameter>
    <parameter id="correlated_params" constant="true">
<arrays:listOfDimensions>
  <arrays:dimension id="i" lowerLimit="1" upperLimit="2" />
</arrays:listOfDimensions>
    </parameter>
  </listOfParameters>
  <listOfInitialAssignments>
    <initialAssignment symbol="V_pop">
<math xmlns="http://www.w3.org/1998/Math/MathML"
      xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
  <apply>
    <cn>105</cn>
  </apply>
</math>
    </initialAssignment>
    <initialAssignment symbol="V_omega">
<math xmlns="http://www.w3.org/1998/Math/MathML"
      xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
  <apply>
    <cn>0.70</cn>
  </apply>
</math>
    </initialAssignment>
    <initialAssignment symbol="Cl_pop">
<math xmlns="http://www.w3.org/1998/Math/MathML"
      xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
  <apply>
    <cn>73</cn>
  </apply>
</math>
    </initialAssignment>
    <initialAssignment symbol="Cl_omega">
<math xmlns="http://www.w3.org/1998/Math/MathML"
      xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
  <apply>
    <cn>0.70</cn>
```

```
  </apply>
</math>
    </initialAssignment>
    <initialAssignment symbol="covariance">
<math xmlns="http://www.w3.org/1998/Math/MathML"
      xmlns:sbml="http://www.sbml.org/sbml/level3/version1/arraymaths/version1">
  <!-- This is an unresolved issue. The l3 V1 Core mathml subset does not support
      matrices. One solution - as above is to use a different MathML subset definition.
  -->
  <matrix>
    <matrixrow>
      <apply><times/><ci>V_omega</ci><ci>V_omega</ci></apply>
      <apply><times/><ci>V_omega</ci><ci>C_omega</ci><ci>V_C_rho</ci></apply>
    </matrixrow>
    <matrixrow>
      <ci>0</ci>
      <apply><times/><ci>C_omega</ci><ci>C_omega</ci></apply>
    </matrixrow>
  </matrix>
</math>
    </initialAssignment>
    <initialAssignment symbol="correlated_means">
<math xmlns="http://www.w3.org/1998/Math/MathML"
      xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
  <vector>
    <ci>V_pop</ci>
    <ci>C_pop</ci>
  </vector>
</math>
    </initialAssignment>
    <initialAssignment symbol="limit_vector">
<math xmlns="http://www.w3.org/1998/Math/MathML"
      xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
  <vector>
    <infinity/>
    <infinity/>
  </vector>
</math>
    </initialAssignment>
    <initialAssignment symbol="correlated_params" >
<math xmlns="http://www.w3.org/1998/Math/MathML"
      xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
  <apply>
    <ci>multivariateNormal</ci><ci>correlated_means</ci><ci>covariance</ci><ci>limitVector</ci><ci>limitVector</ci>
  </apply>
</math>
    </initialAssignment>
    <initialAssignment symbol="Cl">
<math xmlns="http://www.w3.org/1998/Math/MathML"
      xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
  <apply>
    <selector/>
    <ci>correlated_params</ci>
    <cn type="integer">2</cn>
  </apply>
</math>
    </initialAssignment>
    <initialAssignment symbol="V">
<math xmlns="http://www.w3.org/1998/Math/MathML"
      xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
  <apply>
    <selector/>
    <ci>correlated_params</ci>
    <cn type="integer">1</cn>
  </apply>
</math>
    </initialAssignment>
```

```
    </listOfInitialAssignments>
    <-- This is an incomplete model snippet, sufficient to illustrate the use of
    a multivariate distribution. -->
  </model>
</sbml>
```

5.3 User-defined continuous distribution

In this example an additional construct is used to indicate:

- that the MathML within the function definition defines a PDF and so is not a fallback.
- that when invoked a value should be sampled from this PDF.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
      xmlns:distrib="http://www.sbml.org/sbml/level3/version1/distrib/version1"
      distrib:required="true">
  <!-- Code snippet. Not a valid model. -->
  <model id="UserDefined" name="User␣Defined␣Example" substanceUnits="item"
         timeUnits="second" volumeUnits="litre" extentUnits="item">
    <distrib:listOfDistributionsDefns>
  <distrib:explicitPDF distrib:id="mynormal" name="My␣Normal␣Distribution">
        <!-- Function defines a PDF using MathML. The implied behaviour is that
            A value will be sampled from the distribution described by this PDF -->
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <lambda>
            <bvar><ci>mu</ci></bvar>
            <bvar><ci>omega</ci></bvar>
            <apply>
              <apply>
                <times />
                <apply>
                  <power />
                  <exponentiale />
                  <apply>
                    <times />
                    <apply>
                      <times />
                      <cn type='integer'>-1</cn>
                      <cn type='rational'>1<sep />2</cn>
                    </apply>
                    <apply>
                      <power />
                      <apply>
                        <times />
                        <apply>
                          <plus />
                          <ci>x</ci>
                          <apply>
                            <times />
                            <cn type='integer'>-1</cn>
                            <ci>mu</ci>
                          </apply>
                        </apply>
                        <apply>
                          <power />
                          <ci>sigma</ci>
                          <cn type='integer'>-1</cn>
                        </apply>
                      </apply>
                      <cn type='integer'>2</cn>
                    </apply>
                  </apply>
```

```
                </apply>
                <apply>
                  <power />
                  <apply>
                    <times />
                    <ci>omega</ci>
                    <apply>
                      <ci>sqrt</ci>
                      <apply>
                        <times />
                        <cn type='integer'>2</cn>
                        <pi/>
                      </apply>
                    </apply>
                  </apply>
                  <cn type='integer'>-1</cn>
                </apply>
              </apply>
            </apply>
          </lambda>
        </math>
  </distrib:explicitPDF>
    </distrib:listOfDistributionsDefns>
    <listOfParameters>
      <parameter id="V" constant="true">
  <distrib:ref idref="mynormal">
    <distrib:arg idref="mu">
      <distrib:varref idref="V"/>
    </distrib:arg>
    <distrib:arg idref="omega">
      <distrib:varref idref="V_omega"/>
    </distrib:arg>
  </distrib:ref>
      </parameter>
      <parameter id="V_pop" value="100" constant="true"/>
      <parameter id="V_omega" value="0.25" constant="true"/>
    </listOfParameters>
    <listOfInitialAssignments>
      <initialAssignment symbol="V">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
      <ci>V_pop</ci>
  </math>
  <distrib:ref idref="mynormal">
    <distrib:arg idref="mu">
      <distrib:varref idref="V_pop"/>
    </distrib:arg>
    <distrib:arg idref="omega">
      <distrib:varref idref="V_omega"/>
    </distrib:arg>
  </distrib:ref>
      </initialAssignment>
    </listOfInitialAssignments>
  </model>
</sbml>
```

5.4 User-defined discrete distribution

In this example we don't use any special distrib features, by invoke an externally defined function that constructs a PMF from a matrix of values and their associated probabilities. The example is not biologically meaningful, but illustrates that the matrix contains the values and probabilities possible when throwing two dice. These values are encoded in a matrix, which is passed as a parameter in a function, however, it may be that this information is

? defined using NuML or by referring to an external file.

? There has been very little discussion about how to treat user-defined PMFs so this example should be regarded as

a starting point for discussion rather than a final proposal.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
      xmlns:distrib="http://www.sbml.org/sbml/level3/version1/distrib/version1"
      distrib:required="true"
      xmlns:arrays="http://www.sbml.org/sbml/level3/version1/arrays/version1"
      arrays:required="true">
  <!-- Code snippet. Not a valid model. -->
  <model id="UserDefined" name="User␣Defined␣Example" substanceUnits="item"
         timeUnits="second" volumeUnits="litre" extentUnits="item">
    <listOfFunctionDefinitions>
      <functionDefinition id="myDistribution">
        <distrib:explicitPMF>
          <distrib:category id="c2" name="2">
            <distrib:probability>
              <!-- Defined mathematically as a fraction -->
              <math xmlns="http://www.w3.org/1998/Math/MathML"
                    xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
                <apply><divide/><cn>1</cn><cn>36</cn></apply>
              </math>
            </distrib:probability>
          </distrib:category>
          <distrib:category id="c3" name="3">
            <distrib:probability>
              <!-- Defined as decimal -->
              <math xmlns="http://www.w3.org/1998/Math/MathML"
                    xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
                <apply><divide/><cn>0.05555556</cn></apply>
              </math>
            </distrib:probability>
          </distrib:category>
          <distrib:category id="c4" name="4">
            <distrib:probability>
              <!-- Defined by a reference to a parameter or another variable -->
              <math xmlns="http://www.w3.org/1998/Math/MathML"
                    xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
                <ci>catProb</ci>
              </math>
            </distrib:probability>
          </distrib:explicitPMF>
          <!-- below is a dummy function provided to ensure valid SBML Core
              for software that do not understand distrib. -->
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <lambda>
              <apply>
                <cn>0</cn>
              </apply>
            </lambda>
          </math>
        </functionDefinition>
      </listOfFunctionDefinitions>
      <listOfParameters>
        <parameter id="W" constant="true"/>
        <parameter id="catProb" constant="true"/>
      </listOfParameters>
      <listOfInitialAssignments>
        <initialAssignment symbol="catProb">
          <math xmlns="http://www.w3.org/1998/Math/MathML"
                xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
            <apply><divide/><cn>3</cn><cn>36</cn></apply>
          </math>
        </initialAssignment>
        <initialAssignment symbol="W">
          <math xmlns="http://www.w3.org/1998/Math/MathML"
                xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
```

```
          <apply>
            <ci>myDistribution</ci>
          </apply>
        </math>
      </initialAssignment>
    </listOfInitialAssignments>
  </model>
</sbml>
```

6 Prototype implementations

None as yet.

7 Unresolved issues

1. Should the **explicitPDF** class have a truncation attribute. This would allow one to define a truncated distribution without defining the truncated PDF in the MathML definition. Is this desirable or not?

2. Currently the proposal provides and appendix that clarifies the parameters that should be specified for each UncertML distribution and their type. It also defines whether one or more values are sampled from a given distribution. There was some argument on the list about whether this appendix is required. An alternative would be to use UncertML definitions as-is, and live with any inherent ambiguity. We should make a decision about the approach we want to take for the final version of this proposal.

3. The use of arrays. The package and examples have been written that some distribution definitions will require the use of vector or matrix types that can be provided by the Arrays package. While relying on such types enables us to describe a wide range of probability distributions this will come at the cost of delaying this package's approval until Arrays has been developed and approved. There are two alternative strategies. One is to only support distributions that do not require array types (see appendix A). The other would be to define workarounds within distrib to provide array structures without using the arrays package. The decision at COMBINE was to wait. Do we want to revisit this decision?

4. Currently sampling of distributions in permitted in the event related elements of SBML: EventAssignments, Delays and Priorities. The reason for prohibiting sampling in a rate law (or other circumstances where the model is being continuously evaluated) was because this can make the simulation unstable[11]. It is unclear whether a similar situation could arise while sampling from events. The author feels this need to be looked into further by 'who know'.

5. During the review of this proposal it was questioned whether the use of the term PMF was appropriate for the eponymous distrib class: **userDefinedPMF**. The issue is that a PMF is defined for (numerical) values which a random variable can assume and not general 'categories'. Since we permit non-numerical values to define categories such as eye colour (red, green, blue etc) or sex (M, F) it is argued that the current name is misleading. UncertML uses the term discrete probability for a similar construct that describes this, so perhaps a better class name would be **UserDefinedDiscreteProbability** or just **DiscreteProbability**.

6. Limiting the use of **FunctionDefinition**. In review it was pointed out that the restriction on sampling to InitialAssignent, EventAssignment etc affects MathML validity rules in SBML. Currently a function definition can be 'called' from any MathML block in an SBML document. However, distrib imposes an additional special case. *A function definition that defines a distribution can only be called from MathML, where sampling is permitted.* It is not absolutely clear what the technical implications of this are, but it would appear that to implement distrib support any implementation would be required to modify the MathML validation rules in libSBML. Given, that these rules are not designed to interact with packages this may be technically challenging to implement. The author feels that it would be desirable to find an alternative design that avoids the need to modify the MathML validation rules.

[11] See slide entitled Extrinsic noise: http://sbml.org/images/3/3b/Djw-sbml-hackathon-2010-05-04.pdf.

8 Acknowledgements

Much of the initial concrete work leading to this proposal document was carried out at the Statistical Models Workshop in Hinxton in 2011, which was organised by Nicolas le Novère. A list of participants and recordings of the discussion is available from http://sbml.org/Events/Other_Events/statistical_models_workshop_2011. Before that a lot of the ground work was carried out by Darren Wilkinson who led the discussion on distrib at the Seattle SBML Hackathon and before that Colin Gillespie who wrote an initial proposal back in 2005. The author would also like to thank the participants of the distrib sessions during HARMONY 2012 and COMBINE 2012 for their excellent contributions in helping revising this proposal; Sarah Keating, Maciej Swat and Nicolas le Novère for useful discussions, corrections and review comments; and Mike Hucka for LaTeX advice and the beautiful template upon which this document is based.

A Distributions incorporated from UncertML

The distributions described by UncertML[12] will provide the standard external definition of distributions used by Distributions. The definitions are comprehensive, but have multiple implicit definitions in some cases. For example the mean and variance parameters of the Normal distribution can be each be described as a vector of k values, where each mean, variance pair describes a different probability distribution. However, in distrib we want to define one probability density and so it in necessary to disambiguate definitions where more than one is possible.

The table below describes, for each probability distribution defined in UncertML: the arguments (referred to as parameters in UncertML) it takes, the type of each argument and type of the result obtained when a random value (or values) is sampled from the distribution. The type can be either a scalar (in effect a double because this is the only scalar type supported by SBML), or an array (again of doubles), which can have one or more dimensions.

UncertML Name	**Argument**		**Sampled Result**
	Name	**Type**	
BernouliDistribution	probability	scalar	scalar
BetaDistribution	alpha beta	scalar scalar	scalar
BinomialDistribution	numberOfTrials probabilityOfSuccess	scalar	scalar
CauchyDistribution	location scale	scalar scalar	scalar
ChiSquaredDistribution	degreesOfFreedom	scalar	scalar
DirichletDistribution	concentration	array[k], $k \geq 2$	array[k]
ExponentialDistribution	rate	scalar	scalar
FDistribution	numerator denominator	scalar scalar	scalar
GammaDistribution	shape scale	scalar scalar	scalar
GeometricDistribution	probability	scalar	scalar
HypergeometricDistribution	populationSize numberOfTrials numberOfSuccesses	scalar scalar scalar	scalar
InverseGammaDistribution	shape scale	scalar scalar	scalar
LaplaceDistribution	location scale	scalar scalar	scalar
LogNormalDistribution	logScale shape	scalar scalar	scalar
LogisticDistribution	location scale	scalar scalar	scalar
MixtureModel	weight	?	?
MultinomialDistribution	numberofTrials probabiltyOfSuccess	scalar array[k]	array[k]

Continued on next page...

[12] These distributions are defined here: http://www.uncertml.org/distributions

UncertML Name	Argument		Sampled Result
	Name	Type	
MultivariateNormalDistribution	mean covarianceMatrix	array[k] array[k][k]	array[k]
MultivariateStudentTDistribution	mean covarianceMatrix degreesOfFreedom	array[k] array[k][k] scalar	array[k]
NegativeBinomialDistribution	numberOfFailures probability	scalar scalar	scalar
NormalDistribution	mean variance	scalar scalar	scalar
NormalInverseGammaDistribution	mean varianceScaling shape scale	scalar	scalar
ParetoDistribution	scale shape	scalar scalar	scalar
PoissionDistribution	rate	scalar	scalar
StudentTDistribution	location scale degreesOfFreedom	scalar scalar scalar	scalar
WeibullDistribution	scale shape	scalar scalar	scalar
WishartDistribution	scaleMatrix degreesOfFreedom	scalar scalar	scalar

References

Ausbrooks, R., Buswell, S., Carlisle, D., Dalmas, S., Devitt, S., Diaz, A., Froumentin, M., Hunter, R., Ion, P., Kohlhase, M., Miner, R., Poppelier, N., Smith, B., Soiffer, N., Sutor, R., and Watt, S. (2003). Mathematical Markup Language (MathML) Version 2.0 (Second Edition). Available online via the Worldwide Web at http://www.w3.org/TR/MathML/.

Hucka, M., Bergmann, F. T., Hoops, S., Keating, S. M., Sahle, S., Schaff, J. C., Smith, L. P., and Wilkinson, D. J. (2010). The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core. Available via the World Wide Web at http://sbml.org/Documents/Specifications.

SBML Editorial Board (2012). SBML development process for SBML level 3. Available via the World Wide Web at http://sbml.org/Documents/SBML_Development_Process/SBML_Development_Process_for_SBML_Level_3.