Drug Disease Model Resources

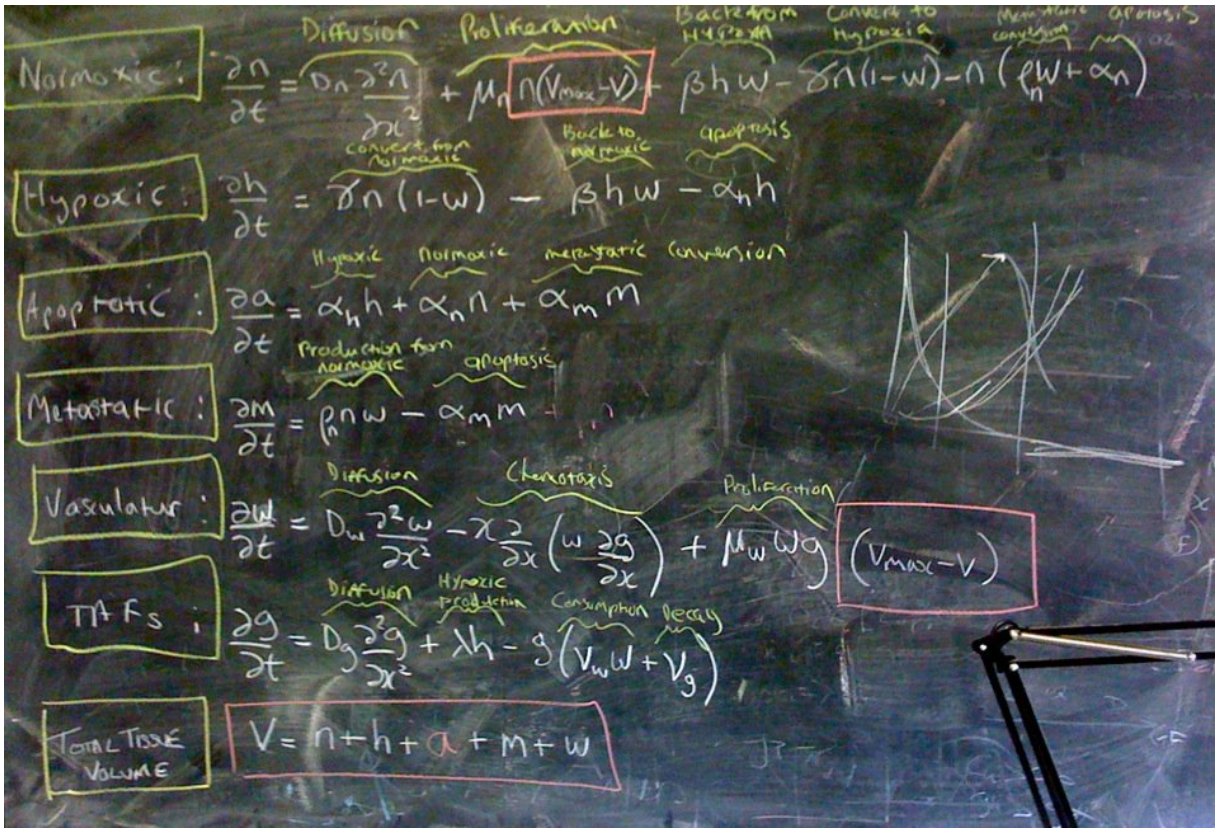
ddmore

Internal Release

Extensions in PharmML 0.9 – DRAFT

Authors:
Maciej J Swat
Florent Yvon
Niels Rode Kristensen

with contributions from:
Gunnar Yngman
Rikard Nordgren
Stuart Moodie
Mike Smith



July 20, 2016

Contents

1 Overview **2**
1.1 Summary of changes/extensions in 0.9 . . . 2

2 Extensions and Changes **5**
2.1 General . . . 5
2.1.1 Box-Cox 2 . . . 5
2.1.2 User-defined transformations . . . 5
2.1.3 User-defined distributions . . . 7
2.1.4 Accessing ODE initial condition values . . . 8
2.1.5 Selecting vector/matrix elements . . . 9
2.1.6 Array dimension operators . . . 10
2.2 Mixture models . . . 10
2.2.1 Mixtures of distributions . . . 10
2.2.2 Mixtures of structural models . . . 11
2.2.2.1 WSMM . . . 12
2.2.2.2 BSMM . . . 12
2.2.3 SymbRef assignments extensions . . . 14
2.3 Covariate model . . . 15
2.3.1 Fixing a bug in referencing of covariate categories . . . 15
2.3.2 Categorical covariate building – clustering . . . 15
2.3.3 Latent covariates . . . 16
2.3.4 Covariate exclusion & inclusion criteria . . . 17
2.4 Parameter model . . . 18
2.4.1 Correlation of random effects/parameters . . . 18
2.5 Observation model . . . 19
2.5.1 Multiple models allowed per block . . . 19
2.5.2 Models are allowed in conditionals . . . 20
2.5.3 Missing element restored . . . 22
2.5.4 Removed redundant named parameters . . . 23
2.6 Autocorrelation of residual errors . . . 24
2.6.1 General models . . . 24
2.6.2 Standard time series models . . . 25
2.6.3 Implementation examples . . . 26
2.7 Trial design . . . 28
2.7.1 Conditioning on trial design elements . . . 28
2.7.2 Missing elements added . . . 29
2.7.3 Declaring occasions . . . 30
2.7.4 Design spaces identifiers . . . 31
2.7.5 Covariates in trial design . . . 31
2.7.5.1 Defining covariates . . . 32
2.7.5.2 Referencing covariates . . . 32
2.8 Modelling steps . . . 33
2.8.1 Missing variable assignment added . . . 33
2.8.2 Tool-specific settings . . . 33
2.9 Other changes . . . 34
2.9.1 List of minor changes . . . 34

2.9.2 Dataset – new column types . . . 34

3 Redesign of differential equations 35
3.1 Introduction . . . 35
3.1.1 Proof of concept . . . 35
3.2 Status Quo . . . 35
3.2.1 Limitations . . . 36
3.3 Building blocks . . . 36
3.4 Initial examples . . . 38
3.4.1 Encoding differential equations . . . 38
3.4.2 Encoding new operators . . . 39
3.5 Integration example . . . 41
3.6 ODE examples . . . 42
3.6.1 Basic ODE . . . 42
3.6.2 ODE with an expressions on LHS . . . 42
3.6.3 ODE with differentials on RHS . . . 43
3.7 PDE examples . . . 44
3.7.1 Advection Equation . . . 44
3.7.2 Diffusion Equation . . . 47
3.7.3 Combined Advection/Diffusion Equation . . . 50
3.7.4 Diffusion Equation – Ghaffarizadeh 2016 . . . 52
3.7.5 Heat equation with different boundary conditions . . . 55
3.7.6 Reaction-diffusion system – Gray-Scott Model . . . 58
3.7.7 Capillary-Tissue Exchange: Convention, Permeation, Reaction, and Diffusion . . . 60
3.7.8 Pattern Formation – Schnakenberg system . . . 63
3.7.9 Breast cancer development – Enderling et al. 2007 . . . 65
3.7.10 PharmML code . . . 67

4 New use cases 68
4.1 Handling IOV in optimal design models . . . 68
4.1.1 Implementation options . . . 68
4.1.2 Option 1 . . . 68
4.1.3 Option 2 . . . 69
4.1.4 Option 3 . . . 70
4.1.5 Option 4 . . . 71
4.2 Constraints . . . 72

Chapter 1

Overview

This document describes extensions and changes in PharmML between versions 0.9 & 0.8.1. The evolution of PharmML since the project start are visualised in Figure 1.1.

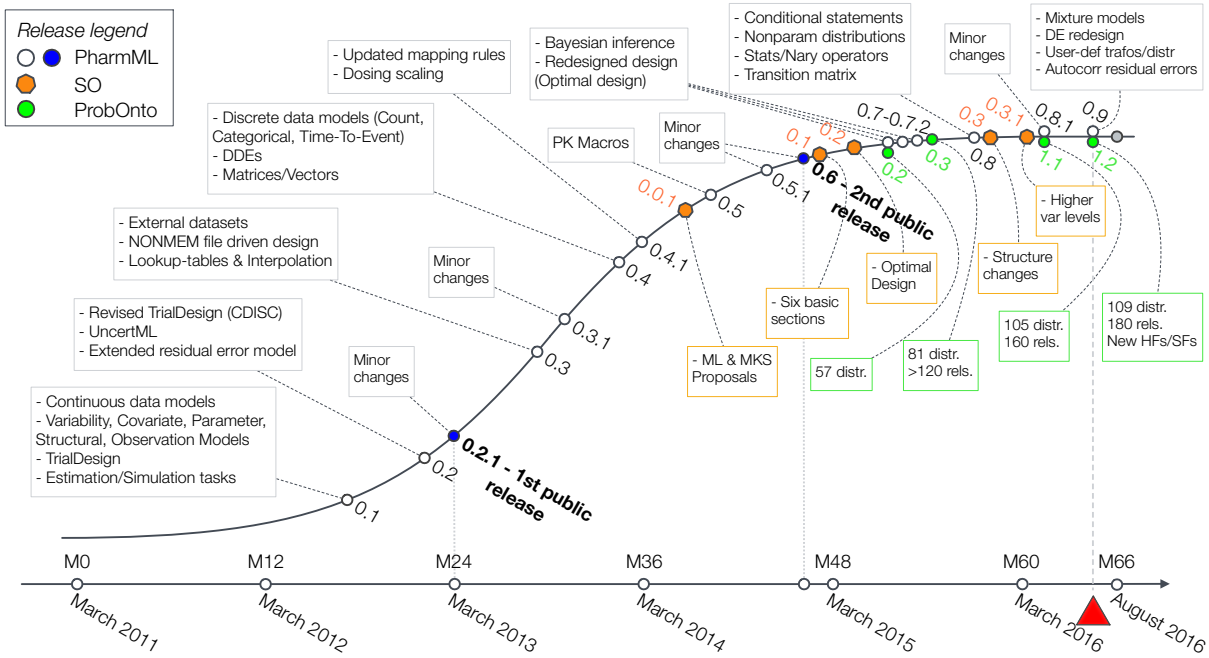


Figure 1.1: PharmML/SO/ProbOnto release history with major extensions listed.

1.1 Summary of changes/extensions in 0.9

The following table summarises the major changes in version 0.9 compared to 0.8.1 described in detail in following chapters.

PharmML element or modelling aspect	version $\leq$ 0.8.1	version 0.9
	General	
Box-Cox transformation Section 2.1.1	standard `BoxCox`	NEW `BoxCox2` version for negative values
User-defined transformations Section 2.1.2	not supported	NEW value *userDefined* for attribute `type`
User-defined distributions Section 2.1.3	not supported	NEW `<UserDefDistribution>` element with attribute `type` w. values CDF, PDF, HF, SF

Constants Section 2.9		imaginary unit i added
Vectors, Matrices Section 2.1.5		NEW `<EndIndex>` element added
ODE initial values Section 2.1.4	not supported	NEW `<InitialValue>` element added
Array dimension operator Section 2.1.6	not supported	NEW `<ArrayDim>` element added with attribute `op` w. values *length, ndims, numel, size*
ConditionalStatement	different namespaces	one namespace `math:` only
	Model Definition	
	Covariate model	
Discrete covariates Section 2.3.2	reference category annotation not supported	NEW `<referenceCategory>` element and the *latent* value of the `type` attribute
Creating new categorical covariates by clustering Section 2.3.2	not supported	referencing possible via `<CatRef>` with additional `symbIdRef` and `blkIdRef` attribute
Latent covariates Section 2.3.3	not supported	NEW `<NumberOfCategories>` element and the *latent* value of the `type` attribute
Exclusion/inclusion criteria Section 2.3.4	not supported	NEW `<Criteria>` element with `type` *exclusive, inclusive* values
Parameter model Section 2.2.3	not supported	Elements `<RandomEffects>`, `<FixedEffect>`, `<RandomVariable1>`, `<RandomVariable2>` accept any expressions
	Structural model	
Calculus operators Section 3.3	not supported	NEW `<VectorCalcOp>` with `op` attribute values *curl, divergence, gradient, laplacian*
ODEs, PDEs Chapter 3	basic support	NEW `<DE>` element with child elements `<BoundaryCondition>`, `<Diff>`, `<DiffVariable>` etc.
	Observation model	
Mixture models Section 2.2.2	not supported	NEW `<MixtureModel>` with child elements `<Proportions>`, `<GroupLabel>` and `<GroupProbabilities>` and attributes `symbId` and `type` with values {*wsmm*, *bsmmSupervised* and *bsmmUnsupervised*}
Multiple models support Section 2.5	allowed only one observation per block	– multiple observations per model supported – conditional declaration supported
Autocorrelation of residual errors, Section 2.6	not supported	NEW `<Autocorrelation>` tag with `type` attribute and child elements `<TimeStepNo>` `<CorrParameters>`
Section 2.5	one observation per block declaration only	multiple observations allowed
Discrete data Section 2.5.4	count data models named parameters available	removed `<IntensityParameter>`, `<DispersionParameter>` etc.
	Trial Design	
Observations Section 2.7.2	missing elements	NEW `<DeltaTime>`, `<BQL>`, `<UQL>`
Covariates declaration Section 2.7.5.1	in top trial design level only	permitted arm-wise as well
Referencing covariates Section 2.7.5	referencing of covariates not supported	referencing of individual covariates and covariate models required with attribute and NEW `<CovariatesReference>` in modelling steps
Referencing occasions Section 2.7.3	not supported	NEW `<OccasionListRef>` element
Design spaces Section 2.7.4	no `oid`'s	– mandatory `<oid>` attributes – referencing of design spaces in optimal

		design with NEW `<DesignSpaceReference>`
Dataset column types Section 2.9.2		NEW `columnType` values *DATE, DAT1 DAT2, DAT3, YTYPE*
Modelling steps		
Tool-specific settings Section 2.8	not supported	NEW `tool` attribute in `<Operation>`
ProbOnto		
Parametric distributions [Swat et al., 2016b]		NEW distributions LogNormal7(μ_N, σ_N), Trapezoidal1(a, b, c, d), HalfNormal2(μ, σ) and WienerDiffusionModel1($\alpha, \beta, \delta, \tau$)
Empirical distributions/ [Swat et al., 2016b]	supported but inconsistent mapping	NEW `<Realisation>` and `<Weight>` elements for declaring and mapping with dataset

Table 1.1: Overview of major differences between versions 0.9 and 0.8.1

Chapter 2

Extensions and Changes

2.1 General

2.1.1 Box-Cox 2

The Box-Cox Transformation, [Box and Cox, 1964], is used to transform data to an approximate normal distribution and the most common version, for $y \geq 0$ was introduced in version 0.7, [Swat et al., 2015a]. Now we introduce also a two-parameter version capable to deal with negative values, which was proposed in the original paper by Box & Cox

$$y^{(\lambda)} = \begin{cases} \dfrac{(y+\lambda_2)^{\lambda_1} - 1}{\lambda_1} & \text{for} \quad \lambda_1 \neq 0 \\ \log(y+\lambda_2) & \text{for} \quad \lambda_1 = 0 \end{cases} \tag{2.1}$$

In PharmML, appliedd to a simple observation model, it reads

```
<Standard symbId="Y">
    <Transformation type="BoxCox2">
        <Parameter>
            <ct:Assign>
                <ct:SymbRef symbIdRef="lambda1"/>
            </ct:Assign>
        </Parameter>
        <Parameter>
            <ct:Assign>
                <ct:SymbRef symbIdRef="lambda2"/>
            </ct:Assign>
        </Parameter>
    </Transformation>
    <Output>
        <ct:SymbRef blkIdRef="sm1" symbIdRef="f"/>
    </Output>
    <ErrorModel>
        <ct:Assign>
            <ct:SymbRef symbIdRef="a"/>
        </ct:Assign>
    </ErrorModel>
    <ResidualError>
        <ct:SymbRef symbIdRef="eps"/>
    </ResidualError>
</Standard>
```

Note that the assumption is that the sequence matters in the code above and that the first and second parameter correspond to λ_1 and λ_2 in eq.(2.1), respectively.

2.1.2 User-defined transformations

There are many transformations one could apply, see e.g. [Sakia, 1992], and to cover all is simply impossible. Therefore, we introduce simple extension mechanism allowing to implement virtually any conceivable transformation and realising already existing features.

The `<Transformation type="...">` element gets

- new value *userDefined* for the `type` attribute

which can be used for such cases. In the following we show how the extended Box-Cox transformation, *Box-Cox 2*, introduced in the previous section, could be implemented using this option.

- it starts with the implementation of the Box-Cox 2 function expressed by the eq.(2.1), called *Box-Cox2Func*, using the `<FunctionDefinition>` structure. The following code snippet shows how this is done.

```
<ct:FunctionDefinition symbId="BoxCox2Func">
    <ct:FunctionArgument symbId="lambda1"/>
    <ct:FunctionArgument symbId="lambda2"/>
    <ct:FunctionArgument symbId="y"/>
    <ct:Definition>
        <ct:Assign>
            <math:Piecewise>
                <!-- omitted Box-Cox 2 formula for brevity -->
            </math:Piecewise>
        </ct:Assign>
    </ct:Definition>
</ct:FunctionDefinition>
```

- this function can then be used to define the following additive observation model applied to the transformed data.

$$h(Cc_{obs}) = h(Cc) + a \times \epsilon$$

with $h \equiv$ 'Box-Cox 2' transformation

```
<ObservationModel blkId="userDefinedTrafoModel">
    <ContinuousData>
        <PopulationParameter symbId="LAMBDA1"/>
        <PopulationParameter symbId="LAMBDA2"/>

        <RandomVariable symbId="eps">
            <Distribution>
                <po:ProbOnto name="StandardNormal1"/>
            </Distribution>
        </RandomVariable>

        <Standard>
            <Transformation type="userDefined">
                <math:FunctionCall>
                    <ct:SymbRef symbIdRef="BoxCox2Func"/>
                    <math:FunctionArgument symbId="lambda1">
                        <ct:SymbRef symbIdRef="LAMBDA1"/>
                    </math:FunctionArgument>
                    <math:FunctionArgument symbId="lambda2">
                        <ct:SymbRef symbIdRef="LAMBDA2"/>
                    </math:FunctionArgument>
                    <math:FunctionArgument symbId="y">
                        <ct:SymbRef blkIdRef="sm1" symbIdRef="Cc"/>
                    </math:FunctionArgument>
                </math:FunctionCall>
            </Transformation>
            <Output>
                <ct:SymbRef blkIdRef="sm1" symbIdRef="Cc"/>
            </Output>
            <ErrorModel>
                <ct:Assign>
                    <ct:SymbRef symbIdRef="a"/>
                </ct:Assign>
            </ErrorModel>
            <ResidualError>
                <ct:SymbRef symbIdRef="eps"/>
            </ResidualError>
        </Standard>

    </ContinuousData>
</ObservationModel>
```

2.1.3 User-defined distributions

Similarly to user-defined functions, there are situations where we might be interested in declarations of distributions not covered by the ProbOnto database, [Swat et al., 2016a]. We start here with two motivating examples and explain later the new supporting elements in PharmML.

Example 1 We consider here a density function describing insulin release, [Grodsky, 1972], which reads

$$f(x) = X_{max} \frac{k\, C\, x^{k-1}}{(C + x^k)^2}$$

As before, section 2.1.2, we define first the density function in question using the `<FunctionDefinition>` element

```
<!-- User-defined density function - Grodsky 1972 -->
<ct:FunctionDefinition symbId="GrodskyPDF">
    <ct:FunctionArgument symbId="Xmax"/>
    <ct:FunctionArgument symbId="k"/>
    <ct:FunctionArgument symbId="C"/>
    <ct:FunctionArgument symbId="x"/>
    <ct:Definition>
        <ct:Assign>
            <!-- we omit the implementation of the following RHS expression -->
            <!-- Xmax * k * C x^{k-1} / (C + x^k)^2 -->
        </ct:Assign>
    </ct:Definition>
</ct:FunctionDefinition>
```

which can then be used to declare a distribution model of a variable, parameter or covariate. Here we use it in an observation model.

```
<ObservationModel blkId="Grodsky1972">
    <ContinuousData>
        <General symbId="Y">
            <Distribution>
                <UserDefDistribution type="PDF">
                    <ct:Assign>
                        <math:FunctionCall>
                            <ct:SymbRef symbIdRef="GrodskyPDF"/>
                            <math:FunctionArgument symbId="Xmax">
                                <ct:SymbRef symbIdRef="Xmax"/>
                            </math:FunctionArgument>
                            <!-- omitted argument mapping for k, C, theta and x -->
                        </math:FunctionCall>
                    </ct:Assign>
                </UserDefDistribution>
            </Distribution>
        </General>
    </ContinuousData>
</ObservationModel>
```

Example 2 Yet another example is a user-defined covariate distribution definition, which reads

```
<Covariate symbId="testCovariate">
    <Categorical>
        <Category catId="cat1"/>
        <Category catId="cat2"/>
        <Distribution>
            <UserDefDistribution type="PMF">
                <ct:Assign>
                    <!-- dummy function -->
                    <ct:Real>1</ct:Real>
                </ct:Assign>
            </UserDefDistribution>
        </Distribution>
    </Categorical>
</Covariate>
```

PharmML support

When declaring a ProbOnto distribution, e.g. the standard normal, it is sufficient to declare its code name, i.e.

```
<Distribution>
    <po:ProbOnto name="StandardNormal1"/>
</Distribution>
```

Instead, as shown in the examples above, we use the introduced `<UserDefDistribution type="...">` element with attribute informing the target tool about the type of the defining function which can be one of the following values $\{CDF, PDF, PMF, HF, SF\}$ as the snippet below illustrates

```
<Distribution>
    <UserDefDistribution type="PDF">
        <ct:Assign>
                <!-- omitted details -->
        </ct:Assign>
    </UserDefDistribution>
</Distribution>
```

Within the `<Assign>` tag, we can either declare the function of interest directly, Example 1, or using `<FunctionCall>`, Example 2, to refer to a previously defined function. Attribute `type` is mandatory.

2.1.4 Accessing ODE initial condition values

It is sometimes desirable to refer or to use the initial condition values of an ODE defined variable as suggested in the NT2MDL converter report [Yngman, 2016]. The following NMTRAN code illustrates one such example

```
A_0(6)=Ccol ; Colistin
IF(A_0(6).EQ.0.04162.AND.TIME.LT.8) KE2 = 0.059
```

and the PharmML implementation reads as follows. The ODE for A6 is shown abbreviated for clarity and only the initial value assignment is fully displayed

```
<StructuralModel blkId="sm1">
    ...
    <ct:DerivativeVariable symbId="A6">
        <ct:Assign>
            <!-- RHS of the ODE omitted -->
        </ct:Assign>
        <ct:InitialCondition>
            <ct:InitialValue>
                <ct:Assign>
                    <ct:SymbRef symbIdRef="Ccol"/>
                </ct:Assign>
            </ct:InitialValue>
        </ct:InitialCondition>
    </ct:DerivativeVariable>
    ...
</StructuralModel>
```

Then the conditional *KE2* parameter assignment reads

```
<ParameterModel blkId="pm1">
    ....
    <IndividualParameter symbId="KE2">
        <ct:Assign>
            <math:Piecewise>
                <math:Piece>
                    <ct:Real>0.059</ct:Real>
                    <math:Condition>
                        <math:LogicBinop op="and">
                            <math:LogicBinop op="eq">
                                <ct:InitialValue>
                                    <ct:Assign>
                                        <ct:SymbRef blkIdRef="sm1" symbIdRef="A6"/>
                                    </ct:Assign>
                                </ct:InitialValue>
                                <ct:Real>0.04162</ct:Real>
                            </math:LogicBinop>
```

```
                        <math:LogicBinop op="lt">
                            <ct:SymbRef symbIdRef="t"/>
                            <ct:Real>8</ct:Real>
                        </math:LogicBinop>
                    </math:LogicBinop>
                </math:Condition>
            </math:Piece>
        </math:Piecewise>
    </ct:Assign>
</IndividualParameter>
...
<ParameterModel blkId="pm1">
```

2.1.5 Selecting vector/matrix elements

New element

- `<EndIndex>`

has been added to `<VectorSelector>` to facilitate efficient selecting operations applied to vectors and matrices. Two examples illustrate its use

Example 1: Picking last element of the 1st column

In this example we assume we don't know the dimension of the matrix but we want to extract the last element of the 1st column of the matrix 'myMatrix'.

```
<PopulationParameter symbId="lastEl">
    <ct:Assign>
        <ct:MatrixSelector>
            <ct:SymbRef symbIdRef="myMatrix"/>
            <ct:Cell>
                <ct:RowIndex>
                    <ct:EndIndex/>
                </ct:RowIndex>
                <ct:ColumnIndex>
                    <ct:Int>1</ct:Int>
                </ct:ColumnIndex>
            </ct:Cell>
        </ct:MatrixSelector>
    </ct:Assign>
</PopulationParameter>
```

Example 2: Assignment of unknown length

We can declare and assign a new vector B with all elements of vector V starting from the 3rd one

$$B = V[3 : end]$$

```
<IndividualParameter symbId="B">
    <ct:Assign>
        <ct:VectorSelector>
            <ct:SymbRef symbIdRef="V"/>
            <ct:Segment>
                <ct:StartIndex>
                    <ct:Int>3</ct:Int>
                </ct:StartIndex>
                <ct:EndIndex/>
            </ct:Segment>
        </ct:VectorSelector>
    </ct:Assign>
</IndividualParameter>
```

2.1.6 Array dimension operators

The following four basic array dimension operator definitions are based on the according Matlab functions `http://nl.mathworks.com/help/matlab/basic-information.html`

- `<ArrayDim op="...">` with following values of `op` attribute
  - `length` – length of array
  - `ndims` – number of array dimensions
  - `numel` – number of array elements
  - `size` – array dimensions

They are applicable to vectors, matrices, sequences, summarized in Table 2.1

Operator	Attribute name	Argument
Largest array dimension length	`length`	`X`
Number of array dimensions	`ndims`	`X`
Number of array elements	`numel`	`X`
Array dimensions	`size`	`X`

Table 2.1: Array dimension operators. `X` can be real vectors, matrices or sequences.

A basic example encoded using the `<ArrayDim op="length">` operator reads

```
<IndividualParameter symbId="langthOfV">
    <ct:Assign>
        <math:ArrayDim op="length">
            <ct:SymbRef symbIdRef="V"/>
        </math:ArrayDim>
    </ct:Assign>
</IndividualParameter>
```

2.2 Mixture models

We consider here mixture models in two variants, see [Lavielle, 2014] for detailed description:

- mixtures of distributions
- mixtures of structural models
  - between-subject model mixtures – each subject belongs to one sub-population
    - supervised
    - unsupervised
  - within-subject model mixtures – there are sub-populations of e.g. cells within subject with different proportions.

2.2.1 Mixtures of distributions

Consider the following mixture model for the volume of distribution assuming two sub-populations with μ_1, σ_1 and μ_2, σ_2, accordingly with mixture parameter $\pi = 0.35$

$$\log(V_i) \sim 0.35\,\mathcal{N}(\log(70), 0.3^2) + 0.65\,\mathcal{N}(\log(42), 0.3^2)$$

It can implemented using the `<MixtureDistribution>` from ProbOnto as the following code snippet shows.

```
<IndividualParameter symbId="V">
    <Distribution>
        <po:ProbOnto name="MixtureDistribution">
            <po:Parameter name="weight">
                <ct:Assign>
```

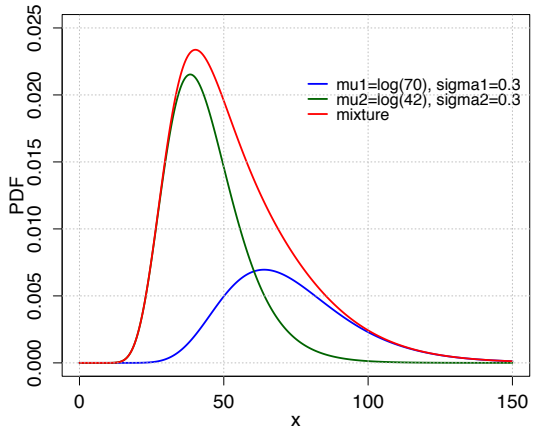


Figure 2.1: PDF plot of the mixture of Log-Normal distribution.

```
                    <ct:SymbRef symbIdRef="pi"/>
                </ct:Assign>
            </po:Parameter>
            <po:MixtureComponent name="LogNormal1">
                <po:Parameter name="meanLog">
                    <ct:Assign>
                        <math:Uniop op="log">
                            <ct:SymbRef symbIdRef="mu1"/>
                        </math:Uniop>
                    </ct:Assign>
                </po:Parameter>
                <po:Parameter name="stdevLog">
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="sigma1"/>
                    </ct:Assign>
                </po:Parameter>
            </po:MixtureComponent>
            <po:MixtureComponent name="LogNormal1">
                <po:Parameter name="meanLog">
                    <ct:Assign>
                        <math:Uniop op="log">
                            <ct:SymbRef symbIdRef="mu2"/>
                        </math:Uniop>
                    </ct:Assign>
                </po:Parameter>
                <po:Parameter name="stdevLog">
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="sigma2"/>
                    </ct:Assign>
                </po:Parameter>
            </po:MixtureComponent>
        </po:ProbOnto>
    </Distribution>
</IndividualParameter>
```

2.2.2 Mixtures of structural models

All mixtures of structural models are encodable using the following new elements

- `<MixtureModel>` with
  - `<StructuralModels>` element which applies in all models along with one of the following elements, dependent on the model type
    - `<Proportions>` for WSMM
    - `<GroupLabel>` for supervised BSMM
    - `<GroupProbabilities>` for unsupervised BSMM
  - attributes

* `symbId` – Id assigned to the mixture model and used later in the actual observation model
* `type` with values *wsmm*, *bsmmSupervised* and *bsmmUnsupervised*

2.2.2.1 WSMM

The between-subject model mixtures reads

$$f = \sum_{m=1}^{M} \pi_{m,i} f_m, \quad \sum \pi_{m,i} = 1$$

and it assumes inter-individual variabilities for the proportions of each model, $\pi_{m,i}$. Its implementation in PharmML happens in the `<ObservationModel>` element

```
<ObservationModel blkId="wsmm">
  <ContinuousData>
    <PopulationParameter symbId="a"/>
    <RandomVariable symbId="eps">
      <Distribution>
        <po:ProbOnto name="StandardNormal1"/>
      </Distribution>
    </RandomVariable>

    <MixtureModel symbId="f" type="wsmm">
      <StructuralModels>
        <Model>
          <ct:Assign>
            <ct:SymbRef blkIdRef="sm1" symbIdRef="f1"/>
          </ct:Assign>
        </Model>
        <!-- omitted f2 & f3 -->
      </StructuralModels>
      <Proportions>
        <Proportion>
          <ct:Assign>
            <ct:SymbRef blkIdRef="pm1" symbIdRef="pi1"/>
          </ct:Assign>
        </Proportion>
        <!-- omitted pi2 & pi3 -->
      </Proportions>
    </MixtureModel>
```

and f can now be used to define the observation model, here one with a constant error model

```
    <Standard symbId="Yobs">
      <Transformation type="identity"/>
      <Output>
        <ct:SymbRef symbIdRef="f"/>
      </Output>
      <ErrorModel>
        <ct:Assign>
          <ct:SymbRef symbIdRef="a"/>
        </ct:Assign>
      </ErrorModel>
      <ResidualError>
        <ct:SymbRef symbIdRef="eps"/>
      </ResidualError>
    </Standard>

  </ContinuousData>
</ObservationModel>
```

2.2.2.2 BSMM

$$f = \sum_{m=1}^{M} \mathbb{1}_{m=z} f_m$$

Supervised

In this case, the group membership is known and the regressor variable z is used to identify it. E.g. in MLXTRAN it is used to condition on and to select the appropriate structural model as the following code snippet shows:

```
g = {use=regressor}
EQUATION:
if g==1
       f = A1
else
       f = A2*exp(-k*max(t,0))
end
```

while in PharmML the previously introduced elements are used

```
<ObservationModel blkId="bsmm">
    <ContinuousData>
        <PopulationParameter symbId="a"/>
        <ct:Variable symbId="z" regressor="yes"/>

        <RandomVariable symbId="eps">
            <Distribution>
                <po:ProbOnto name="StandardNormal1"/>
            </Distribution>
        </RandomVariable>

        <MixtureModel symbId="f" type="bsmmSupervised">
            <StructuralModels>
                <Model>
                    <ct:Assign>
                        <ct:SymbRef blkIdRef="sm1" symbIdRef="f1"/>
                    </ct:Assign>
                </Model>
                <!-- omitted F2 -->
                <Model>
                    <ct:Assign>
                        <ct:SymbRef blkIdRef="sm1" symbIdRef="f3"/>
                    </ct:Assign>
                </Model>
            </StructuralModels>
            <GroupLabel>
                <ct:Assign>
                    <ct:SymbRef symbIdRef="z"/>
                </ct:Assign>
            </GroupLabel>
        </MixtureModel>

        <Standard symbId="Yobs">
               <!-- omitted for brevity -->
        </Standard>

    </ContinuousData>
</ObservationModel>
```

Unsupervised

In this case, the group membership is not known and its proportion, expressed by the population parameter p, in the population is to be estimated.

MLXTRAN code for this model reads:

```
EQUATION:
f1 = A1
f2 = A2*exp(-k*max(t,0))
f=bsmm(f1, p, f2, 1-p)
```

The following PharmML implementation handles the BSMM model only, the implementation of the structural models f_1 and f_2, encoded in `<StructuralModel> sm1`, are not shown.

```
<ObservationModel blkId="bsmm2">
    <ContinuousData>
        <PopulationParameter symbId="a"/>
        <RandomVariable symbId="eps">
            <Distribution>
                <po:ProbOnto name="StandardNormal1"/>
            </Distribution>
        </RandomVariable>

        <MixtureModel symbId="f" type="bsmmUnsupervised">
            <StructuralModels>
                <Model>
                    <ct:Assign>
                        <ct:SymbRef blkIdRef="sm1" symbIdRef="f1"/>
                    </ct:Assign>
                </Model>
                <Model>
                    <ct:Assign>
                        <ct:SymbRef blkIdRef="sm1" symbIdRef="f2"/>
                    </ct:Assign>
                </Model>
            </StructuralModels>
            <GroupProbabilities>
                <GroupProbability>
                    <ct:Assign>
                        <ct:SymbRef blkIdRef="pm1" symbIdRef="p"/>
                    </ct:Assign>
                </GroupProbability>
                <GroupProbability>
                    <ct:Assign>
                        <math:Binop op="minus">
                            <ct:Real>1</ct:Real>
                            <ct:SymbRef blkIdRef="pm1" symbIdRef="p"/>
                        </math:Binop>
                    </ct:Assign>
                </GroupProbability>
            </GroupProbabilities>
        </MixtureModel>

        <Standard symbId="Yobs">
            <!-- Yobs = f + a*eps; omitted for brevity -->
        </Standard>

    </ContinuousData>
</ObservationModel>
```

2.2.3 SymbRef assignments extensions

In a number of cases `<SymbRef>` was the only assignment option. This obviously restrictive structure has been extended to a general assignment using `<Assign>` allowing for virtually any expression. Elements where the new possibility is present are e.g.

- `<RandomVariable1>`, `<RandomVariable2>` – used in the correlation of random effects element
- `<FixedEffect>` – assignment of fixed effects in structured parameter model
- `<RandomEffect>` – assignment of random effects in structured parameter.

Example 1 Element `<RandomEffects>` accept now any assignment, e.g. sum of two fixed effects

```
<FixedEffect>
    <ct:Assign>
        <math:Binop op="plus">
            <ct:SymbRef blkIdRef="pm" symbIdRef="BETA_WT"/>
            <ct:SymbRef blkIdRef="pm" symbIdRef="BETA_CL"/>
        </math:Binop>
    </ct:Assign>
</FixedEffect>
```

Example 2 It is possible to assign a specific vector/matrix element to `<RandomEffects>`, e.g.

```
<RandomEffects>
    <ct:Assign>
        <ct:VectorSelector>
            <ct:SymbRef blkIdRef="pm" symbIdRef="ETA_CL_V_KA"/>
            <ct:Cell>
                <ct:Assign>
                    <ct:Int>1</ct:Int>
                </ct:Assign>
            </ct:Cell>
        </ct:VectorSelector>
    </ct:Assign>
</RandomEffects>
```

2.3 Covariate model

2.3.1 Fixing a bug in referencing of covariate categories

A defined categorical covariate, e.g. sex with two categories F and M

```
<CovariateModel blkId="cm1">
    <Covariate symbId="Sex">
        <Categorical>
            <Category catId="F"/>
            <Category catId="M"/>
        </Categorical>
    </Covariate>
</CovariateModel>
```

is usually used in the parameter model, `<StructuredModel>`, i.e. within the `<FixedEffect>` element as shown in Table 2.2 (left). The use of the `<Category>` however is inconsistent as it was designed to define new categories via the `catId` attribute rather then to refer to existing ones as in this case.

Therefore, when refering to an existing categories, one should use the `<CatRef>` element, see Table 2.2 (right).

support in $\leq$ 0.8.1

```
<IndividualParameter symbId="V">
    <StructuredModel>
        <Transformation type="log"/>
        <LinearCovariate>
            ...
            <Covariate>
                <ct:SymbRef symbIdRef="Sex"/>
                <FixedEffect>
                    <ct:SymbRef symbIdRef="beta_V"/>
                    <Category catId="F"/>
                </FixedEffect>
            </Covariate>
        </LinearCovariate>
        ...
    </StructuredModel>
</IndividualParameter>
```

support in v0.9

```
<IndividualParameter symbId="V">
    <StructuredModel>
        <Transformation type="log"/>
        <LinearCovariate>
            ...
            <Covariate>
                <ct:SymbRef symbIdRef="Sex"/>
                <FixedEffect>
                    <ct:SymbRef symbIdRef="beta_V"/>
                    <ct:CatRef catIdRef="F"/>
                </FixedEffect>
            </Covariate>
        </LinearCovariate>
        ...
    </StructuredModel>
</IndividualParameter>
```

Table 2.2: Referencing of categories of categorical covariates in structured parameter model. Instead of `<Category>` with `catId` attribute, the `<CatRef>` element should be used. For brevity, the block reference `blkIdRef="cm1"` has been removed along with `<PopulationValue>` and `<RandomEffects>` elements.

2.3.2 Categorical covariate building – clustering

Based on other existing covariates, new categorical covariates can be now declared using the so called 'clustering', a mechanism allowing for grouping of categories to define new, more general ones.

New elements and extensions

- boolean reference category, `referenceCategory`, attribute

- `<CatRef>` can be used directly to build new categories by referring to existing ones
- `<CatRef>` equipped with additional `symbIdRef` and `blkIdRef` attributes
- the general assignment via `<Assign>` is also available.

Example

Building new covariate, via clustering of categories of a base covariate, following an MXTRAN example

```
VARIABLES:
    t_GENOTYPE = transform(
           GENOTYPE,
           G1 = {1} reference,
           G2 = {2,3,4}
           ) [use=cov, type=cat]
```

where a transformed covariate 't_GENOTYPE' is declared based on the 'GENOTYPE' covariate.

PharmML implementation reads

- formulate the base covariate as

```
<Covariate symbId="GENOTYPE">
   <Categorical>
      <Category catId="GENOTYPE_cat_1"/>
      <Category catId="GENOTYPE_cat_2"/>
      <Category catId="GENOTYPE_cat_3"/>
      <Category catId="GENOTYPE_cat_4"/>
   </Categorical>
</Covariate>
```

- and declare the new one out of the categories of the *GENOTYPE* covariate

```
<Covariate symbId="t_GENOTYPE">
   <Categorical>
      <Category catId="G1" referenceCategory="true">
         <!-- blkIdRef="cm2" could be used additionally if the reference covariate was -->
         <!-- declared in a different covariate model, "cm2" -->
         <ct:CatRef symbIdRef="GENOTYPE" catIdRef="GENOTYPE_cat_1"/>
      </Category>
      <Category catId="G2">
         <ct:CatRef symbIdRef="GENOTYPE" catIdRef="GENOTYPE_cat_2"/>
         <ct:CatRef symbIdRef="GENOTYPE" catIdRef="GENOTYPE_cat_3"/>
         <ct:CatRef symbIdRef="GENOTYPE" catIdRef="GENOTYPE_cat_4"/>
      </Category>
   </Categorical>
</Covariate>
```

Note that we use the `<ct:CatRef>` element to declare a new category together with the `symbIdRef` attribute pointing to the reference covariate. Additionally `blkIdRef` can be used if the reference covariate was declared in another covariate model.

2.3.3 Latent covariates

Adding *latent* to the covariate type attribute allows to declare any number of latent categorical covariates with any number of categories. The implementation is performed using the

- new value, *latent*, of the `type` attribute
- and the `<NumberOfCategories>` element

as the following code snippet illustrates. We consider a case when the latent covariate has two categories which can be treated as population values to be estimated.

```
<Covariate type="latent" symbId="lCat">
    <Categorical>
        <Category catId="lcat1">
            <ct:Assign>
                <ct:SymbRef blkIdRef="pm1" symbIdRef="plcat1"/>
            </ct:Assign>
        </Category>
        <Category catId="lcat2">
            <ct:Assign>
                <ct:SymbRef blkIdRef="pm1" symbIdRef="plcat2"/>
            </ct:Assign>
        </Category>
        <NumberOfCategories>
            <ct:Assign>
                <ct:Real>2</ct:Real>
            </ct:Assign>
        </NumberOfCategories>
    </Categorical>
</Covariate>
```

Note that e.g. Monolix has a naming convention for latent covariates and their categories in which case the specification of the number of categories for such covariate only is sufficient.

2.3.4 Covariate exclusion & inclusion criteria

This version comes with another extension in handling of covariates, i.e. possibility to declare (any number of)

- exclusion and/or
- inclusion

criteria on continous covariates, e.g. inclusion of subject with BMI between 20 and 25

```
<Covariate symbId="BMI">
    <Continuous>
        <Criteria type="inclusive">
            <math:LogicBinop op="and">
                <math:LogicBinop op="geq">
                    <ct:SymbRef symbIdRef="BMI"/>
                    <ct:Real>20</ct:Real>
                </math:LogicBinop>
                <math:LogicBinop op="leq">
                    <ct:SymbRef symbIdRef="BMI"/>
                    <ct:Real>25</ct:Real>
                </math:LogicBinop>
            </math:LogicBinop>
        </Criteria>
    </Continuous>
</Covariate>
```

and categorical covariates, e.g. exclusion of female subjects

```
<Covariate symbId="Sex">
    <Categorical>
        <Category catId="F"/>
        <Category catId="M"/>
        <Criteria type="exclusive">
            <math:LogicBinop op="eq">
                <ct:SymbRef symbIdRef="Sex"/>
                <ct:CatRef catIdRef="F"/>
            </math:LogicBinop>
        </Criteria>
    </Categorical>
</Covariate>
```

as the code snippets illustrate. They can be applied in all types of tasks but might be especially of interest in simulation and optimal design oriented models.

2.4 Parameter model

2.4.1 Correlation of random effects/parameters

Consider a model with following parameters

$$\log(a) \sim \mathcal{N}(\log(a_{pop}), \omega_a^2)$$
$$b \sim \mathcal{N}(b_{pop}), \omega_b^2)$$
$$\text{logit}(c) \sim \mathcal{N}(\text{logit}(c_{pop}), \omega_c^2)$$

with correlation matrix for the vector ($\log(a)$, b, $\text{logit}(c)$) which reads

$$R = \begin{bmatrix} 1 & r_{ab} & r_{bc} \\ r_{bc} & 1 & r_{bc} \\ r_{bc} & r_{bc} & 1 \end{bmatrix}$$

There are two new ways to encode the above correlation structure

- Option 1: Pairwise correlation for transformed parameters, here one pair is shown only

  $$corr(\log(a), \text{logit}(c)) = r_{ac}$$

  reads in PharmML

```
<Correlation>
    <ct:VariabilityReference>
        <ct:SymbRef blkIdRef="vm1" symbIdRef="indiv"/>
    </ct:VariabilityReference>
    <Pairwise>
        <RandomVariable1>
            <ct:Assign>
                <math:Uniop op="log">
                    <ct:SymbRef symbIdRef="a"/>
                </math:Uniop>
            </ct:Assign>
        </RandomVariable1>
        <RandomVariable2>
            <ct:Assign>
                <math:Uniop op="log">
                    <ct:SymbRef symbIdRef="c"/>
                </math:Uniop>
            </ct:Assign>
        </RandomVariable2>
        <CorrelationCoefficient>
            <ct:Assign>
                <ct:SymbRef symbIdRef="r_ac"/>
            </ct:Assign>
        </CorrelationCoefficient>
    </Pairwise>
</Correlation>
```

- Option 2: Matrix correlation, R, for entire parameter vector

  $$(\log(a), b, \text{logit}(c))$$

  using the `<Correlation>` element notation extended with the

  – standard `<Assign>` element, preceding the `<Matrix>` and `<Pairwise>` elements, to encode the parameter vector capable to hold any combination of transformed or untransformed parameters

  as the following code snippet demonstrates

```
<Correlation deviationMatrixType="CorrMatrix">
    <ct:VariabilityReference>
        <ct:SymbRef blkIdRef="vm1" symbIdRef="indiv"/>
    </ct:VariabilityReference>
    <ct:Assign>
        <ct:Vector>
```

```
                        <ct:VectorElements>
                            <ct:Assign>
                                <math:Uniop op="log">
                                    <ct:SymbRef symbIdRef="a"/>
                                </math:Uniop>
                            </ct:Assign>
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="b"/>
                            </ct:Assign>
                            <ct:Assign>
                                <math:Uniop op="logit">
                                    <ct:SymbRef symbIdRef="c"/>
                                </math:Uniop>
                            </ct:Assign>
                        </ct:VectorElements>
                    </ct:Vector>
                </ct:Assign>
                <Matrix matrixType="Symmetric">
                    <ct:MatrixRow>
                        <ct:Real>1</ct:Real>
                    </ct:MatrixRow>
                    <ct:MatrixRow>
                        <ct:SymbRef symbIdRef="r_ab"/><ct:Real>1</ct:Real>
                    </ct:MatrixRow>
                    <ct:MatrixRow>
                        <ct:SymbRef symbIdRef="r_ac"/><ct:SymbRef symbIdRef="r_bc"/><ct:Real>1</ct:Real>
                    </ct:MatrixRow>
                </Matrix>
            </Correlation>
```

2.5 Observation model

2.5.1 Multiple models allowed per block

Consider continuous PK/PD model implemented as "example1_NONMEM.xml" in the release pack. Instead of declaring separate blocks, the PK and PD models can be now combined within one `<ObservationModel>`. Note, that the data mapping needs to be accordingly amended and parameter a, used in both models, has been duplicated as $a1$ and $a2$.

$$E_{obs} = E + a_1\ \epsilon_E$$
$$Cc_{obs} = Cc + (a_2 + b\ Cc)\ \epsilon_{Cc}$$

```
<ObservationModel blkId="om1">
    <ContinuousData>
        <PopulationParameter symbId="a1"/>
        <PopulationParameter symbId="a2"/>
        <PopulationParameter symbId="b"/>

        <RandomVariable symbId="epsilon_Cc">
            <!-- omitted details -->
        </RandomVariable>
        <RandomVariable symbId="epsilon_E">
            <!-- omitted details -->
        </RandomVariable>

        <!-- FIRST MODEL -->
        <General symbId="E_obs">
            <ct:Assign>
                <math:Binop op="plus">
                    <ct:SymbRef blkIdRef="sm1" symbIdRef="E"/>
                    <math:Binop op="times">
                        <math:FunctionCall>
                            <ct:SymbRef symbIdRef="constantErrorModel"/>
                            <!-- omitted arguments -->
                        </math:FunctionCall>
                        <ct:SymbRef symbIdRef="epsilon_E"/>
```

```
                    </math:Binop>
                </math:Binop>
            </ct:Assign>
        </General>

        <!-- SECOND MODEL - this was not allowed previously -->
        <Standard symbId="Cc_obs">
            <Output>
                <ct:SymbRef blkIdRef="sm1" symbIdRef="Cc"/>
            </Output>
            <ErrorModel>
                <ct:Assign>
                    <math:FunctionCall>
                        <ct:SymbRef symbIdRef="combinedErrorModel"/>
                        <!-- omitted arguments -->
                    </math:FunctionCall>
                </ct:Assign>
            </ErrorModel>
            <ResidualError>
                <ct:SymbRef symbIdRef="epsilon_Cc"/>
            </ResidualError>
        </Standard>

    </ContinuousData>
</ObservationModel>
```

2.5.2 Models are allowed in conditionals

Example 1

Different model types dependent on FLAG value can be declared, e.g.

$$\begin{cases} Cc_{obs} = Cc + a_1\,\epsilon_{Cc} & \text{for} \quad \text{FLAG} == 1 \\ Cc_{obs} = Cc + (a_2 + b\,Cc)\,\epsilon_E & \text{for} \quad \text{FLAG} == 2 \end{cases}$$

```
<ObservationModel blkId="om2">
    <ContinuousData>
        <PopulationParameter symbId="a1"/>
        <PopulationParameter symbId="a2"/>
        <PopulationParameter symbId="b"/>

        <ct:Variable regressor="yes" symbId="FLAG"/>

        <RandomVariable symbId="epsilon_Cc">
            <!-- omitted -->
        </RandomVariable>
        <RandomVariable symbId="epsilon_E">
            <!-- omitted -->
        </RandomVariable>

        <ct:Variable symbId="Cc_obs"/>

        <math:ConditionalStatement>
            <!-- FIRST CONDITION -->
            <math:If>
                <math:Condition>
                    <math:LogicBinop op="eq">
                        <ct:SymbRef symbIdRef="FLAG"/>
                        <ct:Real>1</ct:Real>
                    </math:LogicBinop>
                </math:Condition>
                <General symbIdRef="Cc_obs">
                    <ct:Assign>
                        <math:Binop op="plus">
                            <ct:SymbRef blkIdRef="sm1" symbIdRef="Cc"/>
                            <math:Binop op="times">
                                <math:FunctionCall>
                                    <ct:SymbRef symbIdRef="constantErrorModel"/>
                                    <!-- omitted arguments -->
```

```
                                </math:FunctionCall>
                                <ct:SymbRef symbIdRef="epsilon_Cc"/>
                            </math:Binop>
                        </math:Binop>
                    </ct:Assign>
                </General>
            </math:If>
            <!-- SECOND CONDITION -->
            <math:ElseIf>
                <math:Condition>
                    <math:LogicBinop op="eq">
                        <ct:SymbRef symbIdRef="FLAG"/>
                        <ct:Real>2</ct:Real>
                    </math:LogicBinop>
                </math:Condition>
                <Standard symbIdRef="Cc_obs">
                    <Output>
                        <ct:SymbRef blkIdRef="sm1" symbIdRef="Cc"/>
                    </Output>
                    <ErrorModel>
                        <ct:Assign>
                            <math:FunctionCall>
                                <ct:SymbRef symbIdRef="combinedErrorModel"/>
                                <!-- omitted arguments -->
                            </math:FunctionCall>
                        </ct:Assign>
                    </ErrorModel>
                    <ResidualError>
                        <ct:SymbRef symbIdRef="epsilon_Cc"/>
                    </ResidualError>
                </Standard>
            </math:ElseIf>
        </math:ConditionalStatement>

    </ContinuousData>
</ObservationModel>
```

Example 2

An alternative version of the previous model with a

- structured and
- distribution

based models, dependent on FLAG value, respectively.

$$\begin{cases} Cc_{obs} = Cc + a\,\epsilon_{Cc} & \text{for} \quad \text{FLAG} == 1 \\ Cc_{obs} \sim \mathcal{N}\big(Cc, (a + b\,Cc)\big) & \text{for} \quad \text{FLAG} == 2 \end{cases}$$

```
<ObservationModel blkId="om3">
    <ContinuousData>
        <PopulationParameter symbId="a1"/>
        <PopulationParameter symbId="a2"/>
        <PopulationParameter symbId="b"/>

        <ct:Variable regressor="yes" symbId="FLAG"/>

        <RandomVariable symbId="epsilon_Cc">
            <!-- omitted -->
        </RandomVariable>

        <ct:Variable symbId="Cc_obs"/>

        <math:ConditionalStatement>
            <math:If>
                <math:Condition>
                    <math:LogicBinop op="eq">
                        <ct:SymbRef symbIdRef="FLAG"/>
```

```
                        <ct:Real>1</ct:Real>
                    </math:LogicBinop>
                </math:Condition>
                <General symbIdRef="Cc_obs">
                    <ct:Assign>
                        <math:Binop op="plus">
                            <ct:SymbRef blkIdRef="sm1" symbIdRef="Cc"/>
                            <math:Binop op="times">
                                <math:FunctionCall>
                                    <ct:SymbRef symbIdRef="constantErrorModel"/>
                                    <!-- omitted arguments -->
                                </math:FunctionCall>
                                <ct:SymbRef symbIdRef="epsilon_Cc"/>
                            </math:Binop>
                        </math:Binop>
                    </ct:Assign>
                </General>
            </math:If>
            <math:ElseIf>
                <math:Condition>
                    <math:LogicBinop op="eq">
                        <ct:SymbRef symbIdRef="FLAG"/>
                        <ct:Real>2</ct:Real>
                    </math:LogicBinop>
                </math:Condition>
                <General symbIdRef="Cc_obs">
                    <Distribution>
                        <po:ProbOnto name="Normal1">
                            <po:Parameter name="mean">
                                <ct:Assign>
                                    <ct:SymbRef blkIdRef="sm1" symbIdRef="Cc"/>
                                </ct:Assign>
                            </po:Parameter>
                            <po:Parameter name="stdev">
                                <ct:Assign>
                                    <math:Binop op="plus">
                                        <ct:SymbRef symbIdRef="a2"/>
                                        <math:Binop op="times">
                                            <ct:SymbRef symbIdRef="b"/>
                                            <ct:SymbRef blkIdRef="sm1" symbIdRef="Cc"/>
                                        </math:Binop>
                                    </math:Binop>
                                </ct:Assign>
                            </po:Parameter>
                        </po:ProbOnto>
                    </Distribution>
                </General>
            </math:ElseIf>
        </math:ConditionalStatement>

    </ContinuousData>
</ObservationModel>
```

2.5.3 Missing element restored

During the refactoring/extension of the observation model in version 0.7-0.7.2 we have lost the left-hand side transformation, h in the formula below, option for the `<General>` model, [Swat et al., 2015b].

$$h(y) = H(...)$$

Models like that below, [Beal, 2001], were still encodable but using the distribution-type notation or the very generic `<AssignStatement>` element, as shown below.

Model definition:

$$\log(y_{ij}) = \log(f_{ij} + M) + \frac{f_{ij}}{f_{ij} + M}\epsilon_{1,ij} + \frac{M}{f_{ij} + M}\epsilon_{2,ij}; \quad var(y_{ij}) = \frac{f_{ij}^2 + M^2}{(f_{ij} + M)^2} \tag{2.2}$$

for uncorrelated $\epsilon_{1,ij}, \epsilon_{2,ij} \sim N(0, 1)$.

$$\log(y_{ij}) \sim \mathcal{N}\Big(\log(f_{ij}+M), \frac{f_{ij}^2+M^2}{(f_{ij}+M)^2}\Big) \tag{2.3}$$

Model (2.2) reads in PharmML

```
<ObservationModel blkId="Beal2001">
    <ContinuousData>
        <PopulationParameter symbId="M"/>
        <ct:Variable symbId="y"/>

        <RandomVariable symbId="eps1">
            <Distribution>
                <po:ProbOnto name="StandardNormal1"/>
            </Distribution>
        </RandomVariable>
        <RandomVariable symbId="eps2">
            <Distribution>
                <po:ProbOnto name="StandardNormal1"/>
            </Distribution>
        </RandomVariable>

        <ct:AssignStatement op="eq">
            <math:Uniop op="log">
                <ct:SymbRef symbIdRef="y"/>
            </math:Uniop>
            <math:Binop op="plus">
                <math:Uniop op="log">
                    <math:Binop op="plus">
                        <ct:SymbRef blkIdRef="sm1" symbIdRef="f"/>
                        <ct:SymbRef symbIdRef="M"/>
                    </math:Binop>
                </math:Uniop>
                <math:Binop op="plus">
                    <!-- () * eps1 -->
                    <math:Binop op="times">
                        <math:Binop op="divide">
                            <ct:SymbRef blkIdRef="sm1" symbIdRef="f"/>
                            <math:Binop op="plus">
                                <ct:SymbRef blkIdRef="sm1" symbIdRef="f"/>
                                <ct:SymbRef symbIdRef="M"/>
                            </math:Binop>
                        </math:Binop>
                        <ct:SymbRef symbIdRef="eps1"/>
                    </math:Binop>
                    <!-- () * eps2 -->
                    <math:Binop op="times">
                        <math:Binop op="divide">
                            <ct:SymbRef symbIdRef="M"/>
                            <math:Binop op="plus">
                                <ct:SymbRef blkIdRef="sm1" symbIdRef="f"/>
                                <ct:SymbRef symbIdRef="M"/>
                            </math:Binop>
                        </math:Binop>
                        <ct:SymbRef symbIdRef="eps2"/>
                    </math:Binop>
                </math:Binop>
            </math:Binop>
        </ct:AssignStatement>
    </ContinuousData>
</ObservationModel>
```

Model (2.3) is not shown but is straightforward to implement using the D-Type parameter model.

2.5.4 Removed redundant named parameters

Following, redundant, named parameters element have been removed from the observation model for count data: `<IntensityParameter>`, `<DispersionParameter>`, `<OverDispersionParameter>`, `<ZeroProbabilityParameter>`, `<MixtureProbabilityParameter>`.

2.6 Autocorrelation of residual errors

Although nearly from the very beginning on the to-do list, support of the autocorrelation of residual errors, is only now available and it comes in many different flavours. A discussion of correlation models, based on which the following support has been designed, can be found in [Davidian, 2009], [Bonate, 2011] or [Lavielle, 2014].

We distinguish between the following correlation models

- independent
- unstructured
- exchangeable/compound symmetric
- one-dependent
- two-dependent
- auto-regressive AR(1)
- exponential
- gaussian

2.6.1 General models

In the following, the index i is used to indicate subject, j to indicate the time point.

Independent model

The correlation matrix of the simplest model reads

$$\Gamma_i = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 \end{pmatrix}$$

Unstructured model

The correlation matrix of this most general model reads

$$\Gamma_i(\alpha) = \begin{pmatrix} 1 & \alpha_{12} & \alpha_{13} & \cdots & \alpha_{1n_i} \\ \alpha_{21} & 1 & \alpha_{23} & \cdots & \alpha_{2n_i} \\ \alpha_{31} & \alpha_{32} & 1 & \cdots & \alpha_{3n_i} \\ \cdots & \cdots & \cdots & \ddots & \vdots \\ \alpha_{n_i1} & \alpha_{n_i2} & \alpha_{n_i3} & \alpha_{n_i,n_{i-1}} & 1 \end{pmatrix}, \quad 0 \leq \alpha_{jj'} \leq 1$$

with $\alpha_{j,j'} = \alpha_{j',j}$. This model is also called *unstructured* as there is no mathematical relation between the level of correlation and index of the time course. In other words, all correlations are uncommon and unique to each subject.

Exchangeable/compound symmetric model

The correlation matrix reads

$$\Gamma_i(\alpha) = \begin{pmatrix} 1 & \alpha & \alpha & \cdots & \alpha \\ \alpha & 1 & \alpha & \cdots & \alpha \\ \alpha & \alpha & 1 & \cdots & \alpha \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha & \alpha & \alpha & \cdots & 1 \end{pmatrix}$$

This model assumes constant correlation between measurements.

2.6.2 Standard time series models

One-dependent model

The correlation matrix reads

$$\Gamma_i(\alpha) = \begin{pmatrix} 1 & \alpha_1 & 0 & \cdots & 0 \\ \alpha_1 & 1 & \alpha_2 & \cdots & 0 \\ 0 & \alpha_2 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \alpha_{n_i-1} & 1 \end{pmatrix}$$

A special case of this model arises if one assumes that $\alpha_j \equiv \alpha$

$$\Gamma_i(\alpha) = \begin{pmatrix} 1 & \alpha & 0 & \cdots & 0 \\ \alpha & 1 & \alpha & \cdots & 0 \\ 0 & \alpha & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \alpha & 1 \end{pmatrix}$$

This model assumes that each observation is correlated with the nearest neighbour only.

Two-dependent model

[Bonate, 2011] describes a model which assumes correlation with two neighbour time points and its matrix reads

$$\Gamma_i(\alpha) = \begin{pmatrix} 1 & \alpha_1 & \alpha_2 & 0 & \cdots & 0 \\ \alpha_1 & 1 & \alpha_1 & \alpha_2 & \cdots & 0 \\ \alpha_2 & \alpha_1 & 1 & \alpha_1 & \cdots & 0 \\ 0 & \alpha_2 & \alpha_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \alpha_2 & \alpha_1 & 1 & \alpha_1 \\ 0 & \cdots & 0 & \alpha_2 & \alpha_1 & 1 \end{pmatrix}$$

Autoregressive model of order 1, AR(1)

This model holds under a number of assumptions, such as equidistant time steps. (For more details see e.g. https://onlinecourses.science.psu.edu/stat510/?q=node/60)

$$\Gamma_i(\alpha) = \begin{pmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n_i-1} \\ \alpha & 1 & \alpha & \alpha^2 & \cdots \\ \alpha^2 & \alpha & 1 & \alpha & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha^{n_i-1} & \cdots & \alpha^2 & \alpha & 1 \end{pmatrix}$$

with $0 \leq \alpha \leq 1$. For an individual the autocorrelation function takes on the form $ACF(k) = \alpha^k$ with k the lag - number of equidistant intervals.

Exponential correlation model

The exponential correlation is represented by the autocorrelation function

$$\rho(k) = \exp(-\alpha k), \quad \alpha > 0$$

from which it follows for correlation between two residual errors $\epsilon_{i,j}$ and $\epsilon_{i,j'}$

$$\text{corr}(\epsilon_{i,j}, \epsilon_{i,j'}) = \exp(-\alpha |t_{i,j} - t_{i,j'}|).$$

This results in a correlation matrix

$$\Gamma_i(\alpha) = \begin{pmatrix} 1 & \alpha_*^{|t_{i1}-t_{i2}|} & \alpha_*^{|t_{i1}-t_{i3}|} & \cdots & \alpha_*^{|t_{i1}-t_{in_i-1}|} \\ & 1 & \alpha_*^{|t_{i2}-t_{i3}|} & \cdots & \alpha_*^{|t_{i2}-t_{in_i-1}|} \\ & & \ddots & \vdots & \vdots \\ & & & 1 & \alpha_*^{|t_{in_i-2}-t_{in_i-1}|} \\ & & & & 1 \end{pmatrix}$$

with $\alpha_* = \exp(-\alpha)$.

It turns out that the AR(1) model is a special case of the exponential one if one considers equidistant time steps in the time series, $|t_{ij} - t_{ij+1}| = const$.

Gaussian correlation model

The Gaussian correlation is represented by the autocorrelation function

$$\rho(k) = \exp(-\alpha k^2), \quad \alpha > 0$$

which yields

$$\text{corr}(\epsilon_{i,j}, \epsilon_{i,j'}) = \exp\{-\alpha(t_{i,j} - t_{i,j'})^2\}.$$

This correlation matrix reads

$$\Gamma_i(\alpha) = \begin{pmatrix} 1 & e^{-\alpha(t_{i1}-t_{i2})^2} & e^{-\alpha(t_{i1}-t_{i3})^2} & \cdots & e^{-\alpha(t_{i1}-t_{in_i-1})^2} \\ & 1 & e^{-\alpha(t_{i2}-t_{i3})^2} & \cdots & e^{-\alpha(t_{i2}-t_{in_i-1})^2} \\ & & \ddots & \vdots & \vdots \\ & & & 1 & e^{-\alpha(t_{in_i-2}-t_{in_i-1})^2} \\ & & & & 1 \end{pmatrix}$$

2.6.3 Implementation examples

The following table summarises the model types and the required elements and their types.

Model type	**1st Argument**	Type	**2nd Argument**	Type
Independent	–	–	–	–
Unstructured	# time points	$n \in N$	$\left(\frac{n^2-n}{2}\right)$ α_{ij}'s	$0 \leq \alpha_{ij} \leq 1$
Exchangeable	# time points	$n \in N$	α	$\alpha \in R$
One-dependent1	# time points	$n \in N$	$(n-1)$ α_i's	$0 \leq \alpha_i \leq 1$
One-dependent2	# time points	$n \in N$	α	$0 \leq \alpha \leq 1$
Two-dependent	# time points	$n \in N$	α_1, α_2	$0 \leq \alpha \leq 1$
AR(1)	# time points	$n \in N$	α	$0 \leq \alpha \leq 1$
Exponential	# time points	$n \in N$	α	$\alpha \in R$
Gaussian	# time points	$n \in N$	α	$\alpha \in R$

Table 2.3: Autocorrelation models – an overview. The first argument is identical for all models, the number of time steps to be considered, except the *Independent* one. The second varies dependent on the model, it's either a scalar or a vector with the required for each model number of elements.

The following structure is provided to encode the autocorrelation models

- `<Autocorrelation type="">`

- `type` – *independent, unstructured, exchangeable, oneDependent1, oneDependent2, twoDependent, AR1, exponential, gaussian*
- `<TimeStepNo>` – (optional) element for specification of the number of time steps to consider
- `<CorrParameters>` – (optional) scalar or vector element with α parameters, see Table 2.3 for details.

Below we list a few application examples for selected models (note the referenced coefficients must be declared before)

- `independent` – single line declaration is sufficient

```
<Autocorrelation type="independent"/>
```

- `exchangeable` – for 5 time points one must declare 10 coefficients, see Table 2.3,

```
<Autocorrelation type="exchangeable"> <!-- 5 steps -->
    <TimeStepNumber>
        <ct:Assign>
            <ct:Int>5</ct:Int>
        </ct:Assign>
    </TimeStepNumber>
    <CorrParameters>
        <ct:Assign>
            <ct:Vector>
                <ct:VectorElements> <!-- (n^2-n)/2 = 10 -->
                    <ct:SymbRef symbIdRef="alpha_12"/>
                    <ct:SymbRef symbIdRef="alpha_13"/>
                    <ct:SymbRef symbIdRef="alpha_14"/>
                    <ct:SymbRef symbIdRef="alpha_15"/>
                    <ct:SymbRef symbIdRef="alpha_23"/>
                    <ct:SymbRef symbIdRef="alpha_24"/>
                    <ct:SymbRef symbIdRef="alpha_25"/>
                    <ct:SymbRef symbIdRef="alpha_34"/>
                    <ct:SymbRef symbIdRef="alpha_35"/>
                    <ct:SymbRef symbIdRef="alpha_45"/>
                </ct:VectorElements>
            </ct:Vector>
        </ct:Assign>
    </CorrParameters>
</Autocorrelation>
```

- `oneDependent1` – 4 coefficients, $\alpha_1, ..., \alpha_4$, are required

```
<Autocorrelation type="oneDependent1">
    <TimeStepNumber>
        <ct:Assign>
            <ct:Int>5</ct:Int>
        </ct:Assign>
    </TimeStepNumber>
    <CorrParameters>
        <ct:Assign>
            <ct:Vector>
                <ct:VectorElements> <!-- n-1 = 4 coefficients -->
                    <ct:SymbRef symbIdRef="alpha_1"/>
                    <ct:SymbRef symbIdRef="alpha_2"/>
                    <ct:SymbRef symbIdRef="alpha_3"/>
                    <ct:SymbRef symbIdRef="alpha_4"/>
                </ct:VectorElements>
            </ct:Vector>
        </ct:Assign>
    </CorrParameters>
</Autocorrelation>
```

- `twoDependent` – 2 coefficients, α_1 and α_2, are required

```
<Autocorrelation type="twoDependent">
    <TimeStepNumber>
        <ct:Assign>
            <ct:Int>5</ct:Int>
        </ct:Assign>
```

```
            </TimeStepNumber>
            <CorrParameters>
                <ct:Assign>
                    <ct:Vector>
                        <ct:VectorElements>
                            <ct:SymbRef symbIdRef="alpha_1"/>
                            <ct:SymbRef symbIdRef="alpha_2"/>
                        </ct:VectorElements>
                    </ct:Vector>
                </ct:Assign>
            </CorrParameters>
        </Autocorrelation>
```

- `exponential` – a single coefficient, α, is required

```
        <Autocorrelation type="exponential">
            <TimeStepNumber>
                <ct:Assign>
                    <ct:Int>5</ct:Int>
                </ct:Assign>
            </TimeStepNumber>
            <CorrParameters>
                <ct:Assign>
                    <ct:SymbRef symbIdRef="alpha"/>
                </ct:Assign>
            </CorrParameters>
        </Autocorrelation>
```

2.7 Trial design

2.7.1 Conditioning on trial design elements

Conditioning of e.g. the covariate distribution on study arm was possible previously, but used an inconsistent grammar

```
        <math:Condition>
            <math:LogicBinop op="eq">
                <math:ArmRef oidRef="arm2"/>
                <ct:True/>
            </math:LogicBinop>
        </math:Condition>
```

i.e. treating arm identifier, here "arm2", as a boolean variable. With the introduction of the `<StudyArm>` (along with other trial design elements, see section 2.9) the implementation is now consistent and reads

```
<CovariateModel oid="CM5">

    <Covariate symbId="W">
        <mdef:Continuous>
            <mdef:Distribution>
                <math:Piecewise>
                    <math:Piece>
                        <po:ProbOnto name="Normal1">
                            <po:Parameter name="mean">
                                <ct:Assign>
                                    <ct:Real>70</ct:Real>
                                </ct:Assign>
                            </po:Parameter>
                            <po:Parameter name="stdev">
                                <ct:Assign>
                                    <ct:Real>13</ct:Real>
                                </ct:Assign>
                            </po:Parameter>
                        </po:ProbOnto>
                        <math:Condition>
                            <math:LogicBinop op="eq">
                                <StudyArm/>
                                <math:ArmRef oidRef="arm1"/>
```

```
                        </math:LogicBinop>
                    </math:Condition>
                </math:Piece>
                <math:Piece>
                    <po:ProbOnto name="Normal1">
                        <po:Parameter name="mean">
                            <ct:Assign>
                                <ct:Real>65</ct:Real>
                            </ct:Assign>
                        </po:Parameter>
                        <po:Parameter name="stdev">
                            <ct:Assign>
                                <ct:Real>10</ct:Real>
                            </ct:Assign>
                        </po:Parameter>
                    </po:ProbOnto>
                    <math:Condition>
                        <math:LogicBinop op="eq">
                            <StudyArm/>
                            <math:ArmRef oidRef="arm2"/>
                        </math:LogicBinop>
                    </math:Condition>
                </math:Piece>
            </math:Piecewise>
        </mdef:Distribution>
    </mdef:Continuous>
</Covariate>
</CovariateModel>
```

2.7.2 Missing elements added

Despite being described in the trial design proposal, [Comets et al., 2015], the following elements were overlooked in the previous schema and are now available, such as

- `<DeltaTime>` – (defaults to 0): the minimum time between two sampling times
- `<BQL>` – (optional): the value of the limit of quantification for this response (assay characteristic), if it can be different in different arms; defaults to none
- `<UQL>` – (optional): upper limit of quantification: defaults to none

As example we consider the following MDL declaration of an observation

```
sampleControl : {type is simple, sampleTime=[8,...,48], outcome = Y, deltaTime=1, BQL=0.1, UQL=10}
```

which reads in PharmML

```
<Observations>
    <Observation oid="sampleControl">
        <ObservationTimes>
            <ct:Assign>
                <ct:Vector>
                    <ct:VectorElements>
                        <ct:Real>8</ct:Real>
                        <!-- omitted values -->
                        <ct:Real>48</ct:Real>
                    </ct:VectorElements>
                </ct:Vector>
            </ct:Assign>
        </ObservationTimes>
        <DeltaTime>
            <ct:Assign>
                <ct:Real>1</ct:Real>
            </ct:Assign>
        </DeltaTime>
        <BQL>
            <ct:Assign>
                <ct:Real>0.1</ct:Real>
            </ct:Assign>
        </BQL>
```

```
            <UQL>
                <ct:Assign>
                    <ct:Real>10</ct:Real>
                </ct:Assign>
            </UQL>
            <!-- omitted details -->
```

2.7.3 Declaring occasions

In version v0.8.1 once the occasions have been usually defined in the top level of the trial design, it was implicitly assumed that they hold for all subsequently defined arms. Even though declaring occasions was possible arm-wise already in the previous version as well what is new is the option to refer explicitly to certain occasion lists using the `<OccasionListRef>` element.

This extension therefore allows to combine flexibly and selectively various occasion levels.

Option 1 Occasions declared within arms

```
<Arms>
    <Arm oid="arm2">
        ...
        <OccasionSequence>
            <OccasionList oid="arm2_OCC">
                <ct:VariabilityReference>
                    <ct:SymbRef blkIdRef="vm_mdl" symbIdRef="OCC"/>
                </ct:VariabilityReference>
                <Occasion oid="arm2_OCC_1">
                    <Start>
                        <ct:Assign>
                            <ct:Int>0</ct:Int>
                        </ct:Assign>
                    </Start>
                </Occasion>
                <Occasion oid="arm2_OCC_2">
                    <Start>
                        <ct:Assign>
                            <ct:Int>120</ct:Int>
                        </ct:Assign>
                    </Start>
                </Occasion>
            </OccasionList>
        </OccasionSequence>
        ...
    </Arm>
</Arms>
```

Option 2 Occasions declared in the top level of the trial design and referenced in the arms

```
<Occasions>
    <OccasionList oid="ol1">
        <ct:VariabilityReference>
            <ct:SymbRef blkIdRef="vm_mdl" symbIdRef="OCC"/>
        </ct:VariabilityReference>
        <Occasion oid="OCC_1">
            <Start>
                <ct:Assign>
                    <ct:Int>0</ct:Int>
                </ct:Assign>
            </Start>
        </Occasion>
        <Occasion oid="OCC_2">
            <Start>
                <ct:Assign>
                    <ct:Int>120</ct:Int>
                </ct:Assign>
            </Start>
        </Occasion>
    </OccasionList>
</Occasions>
```

```
<Arms>
    <Arm oid="arm2">
        ...
        <OccasionSequence>
            <OccasionListRef oidRef="ol1"/>
        </OccasionSequence>
        ...
    </Arm>
</Arms>
```

2.7.4 Design spaces identifiers

Design spaces didn't have identifiers in the previous version, and the rule was that all declared design spaces had to be considered in the modelling steps. This limitation has been now relaxed with

- mandatory object identifier attribute `oid` on the `<DesignSpace>` element

```
<DesignSpace oid="ds1">
    <InterventionRef oidRef="t1"/>
    <DoseAmount>
        <ct:Assign>
            <!-- omitted details -->
        </ct:Assign>
    </DoseAmount>
```

- this `oid` allows us then to reference only the desired design spaces in the `<OptimalDesignStep>` step using the new `<DesignSpaceReference>` element which can take any number of `<OidRef>` tags

```
<OptimalDesignStep oid="evaTask1">

    <DesignSpaceReference>
        <ct:OidRef oidRef="ds1"/>
        <ct:OidRef oidRef="ds2"/>
    </DesignSpaceReference>
```

This feature allows for more flexibility in the set up of the optimal design tasks.

2.7.5 Covariates in trial design

Covariates, either as covariate model or individual covariates, play a very important role in trial design and require comprehensive support for simulation, estimation and optimal design tasks. Usually a base covariate model is declared in the `<ModelDefinition>` section

```
<ModelDefinition>
    ...
    <CovariateModel blkId="cm1">
        <Covariate symbId="TRT">
            <Category catId="A"/>
            <Category catId="B"/>
        </Covariate>
    </CovariateModel>
    ...
</ModelDefinition>
```

and then another one in `<TrialDesign>`

```
<TrialDesign>
    ...
    <Covariates>
        <CovariateModel oid="td_cm">
            <CovariateModelRef oidRef="cm"/>
            <Covariate symbId="TRT">
                <mdef:Categorical/>
            </Covariate>
        </CovariateModel>
    </Covariates>
    ...
</TrialDesign>
```

referring to the former one, `cm1`, possibly adding new features to it and overwriting it. See Figure 2.2 for another possibility when individually stored covariates are mapped the base covariate model .

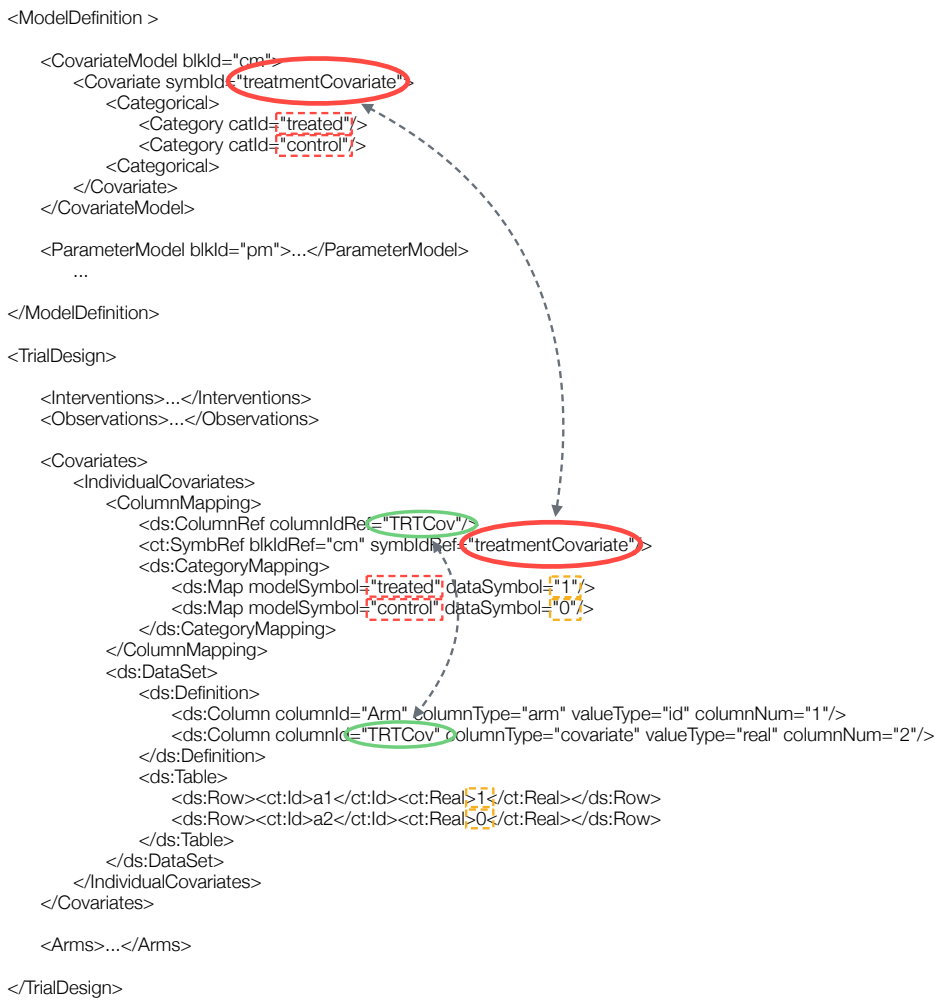


Figure 2.2: An example of how covariate are handled when they are encoded explicitly in the trial design using the `<IndividualCovariates>` element.

2.7.5.1 Defining covariates

In version $\leq$ 0.8.1 declaration of covariates was permitted only in the top level of the trial design, which posed a limitation on models. This restriction is now removed as the following section will show. For more examples, see section 4.1.

2.7.5.2 Referencing covariates

The above general example can be extended to cases when covariates are defined per subject/arm, using the `<IndividualCovariates>` element, and when covariates are associated with all arms or arm-wise.

The first snippet illustrates a case when both a covariate model, `td_cm1`, and individual covariates, `indivCov`, are defined.

```
<TrialDesign>
    <Covariates>
        <CovariateModel oid="td_cm1">
            <!-- ... -->
        </CovariateModel>

        <IndividualCovariates oid="indivCov">
            <!-- omitted mapping -->
            <ds:DataSet>
                <ds:Definition>
                    <ds:Column columnId="ID" columnType="id" valueType="string" columnNum="1"/>
                    <ds:Column columnId="ARM" columnType="arm" valueType="id" columnNum="2"/>
```

```
                    <ds:Column columnId="covariate1" columnType="covariate" valueType="string" columnNum="3"/>
                    <ds:Column columnId="covariate2" columnType="covariate" valueType="string" columnNum="4"/>
                    <!-- omitted other covariates -->
                </ds:Definition>
                <!-- omitted inline covariates records or external file reference -->
            </ds:DataSet>
        </IndividualCovariates>
    </Covariates>
```

These can then be referenced in the design arms in

- the arms top level

```
    <Arms>
        <CovariateModelRef oidRef="td_cm1"/>
        <!-- OR -->
        <IndividualCovariateRef oidRef="indivCov"/>

        <Arm oid="arm1">
            <!-- omitted interventions, observations etc. -->
            <!-- no covariate declarations here needed-->
        </Arm>
        <Arm oid="arm2">
            <!-- omitted interventions, observations etc. -->
            <!-- no covariate declarations here needed-->
        </Arm>
```

- or arm-wise – here we assume further that individual covariates has been defined for each arm separately, as `indivCov1` and `indivCov2`, respectively

```
    <Arms>
        <Arm oid="arm1">
            <IndividualCovariatesRef oidRef="indivCov1"/>
            <!-- OR -->
            <CovariateModelRef oidRef="td_cm1"/> <!-- REF to CovariateModel if no adjustment is required-->

            <CovariateModel oid="cm_arm1">
                <CovariateModelRef oidRef="td_cm1"/>
                <!-- arm 1 specific adjustments to covariate model -->
            </CovariateModel>
        </Arm>
        <Arm oid="arm2">
            <IndividualCovariatesRef oidRef="indivCov2"/>

            <CovariateModel oid="cm_arm2">
                <CovariateModelRef oidRef="td_cm1"/>
                <!-- arm 2 specific adjustments to covariate model -->
            </CovariateModel>

        </Arm>
    </Arms>
</TrialDesign>
```

2.8 Modelling steps

2.8.1 Missing variable assignment added

`<VariableAssignment>` has been added to the optimal design tasks and can now be used to assign values to any model parameter or variable similarly to what was possible in estimation or simulation tasks.

2.8.2 Tool-specific settings

- The declaration of tool specific settings is now possible with new `tool` attribute in `<Operation>`

```
<ModellingSteps xmlns="http://www.pharmml.org/pharmml/0.9/ModellingSteps">

    <!-- omitted elements -->
```

```
            <ParametersToEstimate>
                <ParameterEstimation>
                    <ct:SymbRef symbIdRef="d"/>
                    <InitialEstimate fixed="false">
                        <ct:Real>1</ct:Real>
                    </InitialEstimate>
                </ParameterEstimation>
                <!-- omitted elements -->
            </ParametersToEstimate>

            <Operation tool="Monolix" opType="estPop">
                <Algorithm definition="SAEM"/>
            </Operation>

            <Operation tool="NONMEM" opType="estPop">
                <Algorithm definition="FOCE"/>
            </Operation>

        </EstimationStep>
```

2.9 Other changes

2.9.1 List of minor changes

- new elements `<Action>`, `<Combination>`, `<Intervention>` `<Observation>`, `<Occasion>`, `<StudyArm>` added – allowing for conditioning of these trial design elements if required. See section 2.7.1 for an application example
- `CatRef` element added to operations
- imaginary unit i added – mathematical constant which is the square root of -1
- made order optional in `<Operation>`

2.9.2 Dataset – new column types

To provide the support for Monolix specific column types, following values have can be now assigned to the `columnType` attribute

- *DATE* (or *DAT1*, *DAT2*, *DAT3*) – used to time stamp
- *YTYPE* – response type identifier

see the Monolix specification for details.

Chapter 3

Redesign of differential equations

3.1 Introduction

In this chapter we introduce new notation for differential equations. The goal was to come up with a generic, unified support for encoding of

- ordinary differential equations (ODE) – eliminating limitations of the previous versions and
- delay differential equations (DDE) and
- partial differential equations (PDE) – for models in one spatial and one time variable.

The essential extension offered in this version is the possibility to implement PDEs. Application examples include many models of biological processes and even clinically relevant cases, such as

- Advection – modeling concentrations of materials in capillaries, section 3.7.1
- Capillary-tissue exchange: convention, permeation, reaction, and diffusion, section 3.7.7
- Diffusion models, e.g. section 3.7.4
- Breast cancer development – Enderling et al. 2007, section 3.7.9
- Pattern Formation – Schnakenberg system, section 3.7.8
- Age structured models.

3.1.1 Proof of concept

Modelling software tools such as JSim, [Raymond et al., 2003], XPPAUT, [Ermentrout, 2002], or Matlab, provide a proof of concept for this proposal to support the encoding of PDE model in one spatial x and time t variable.

As shown in sections 3.7.1–3.7.3 and 3.7.8 a large class of relevant PDEs, with initial and boundary conditions, can be encoded in a way which allows their automatic parsing and translations into machine code of a target tool.

3.2 Status Quo

So far the encoding of differential equations (DE) was limited to ODEs and DDEs. The implementation of an ODE, e.g.

$$\frac{dA}{dt} = -kA \quad \text{with} \quad A(t=t0) = A0$$

was done using the `<DerivativeVariable>` element as the following code snippet shows

```
<ct:DerivativeVariable symbId="A">
    <ct:Assign>
        <math:Binop op="times">
            <math:Uniop op="minus">
                <ct:SymbRef blkIdRef="pm1" symbIdRef="k"/>
            </math:Uniop>
            <ct:SymbRef symbIdRef="A"/>
        </math:Binop>
    </ct:Assign>
    <ct:IndependentVariable>
        <ct:SymbRef symbIdRef="t"/>
    </ct:IndependentVariable>
    <ct:InitialCondition>
        <ct:InitialValue>
            <ct:Assign>
                <ct:SymbRef blkIdRef="pm1" symbIdRef="A0"/>
            </ct:Assign>
        </ct:InitialValue>
        <ct:InitialTime>
            <ct:Assign>
                <ct:SymbRef blkIdRef="pm1" symbIdRef="t0"/>
            </ct:Assign>
        </ct:InitialTime>
    </ct:InitialCondition>
</ct:DerivativeVariable>
```

with parameters $k, A0, t0$ encoded in the `<ParameterModel>` (not shown here).

3.2.1 Limitations

This notation had number of drawbacks, e.g. it was impossible to define any DEs with expressions on left-hand side or to have derivatives on right-hand side. The encoding of PDEs was beyond the scope as well. In this release we propose a new format for differential equations free of the limitations.

3.3 Building blocks

This section contain the description and listing of all DE elements and calculus operators described later in multiple examples. Even though the `<Integral>` is not used in DEs, it is part of the required calculus elements and is listed below.

The choice of the calculus operators is based on MathML3[1] but the notation is more explicit and aligned with that used so far in PharmML.

- differential equation – `<DE type="">`
  - `<AssignStatement op="eq">` with left- and right-hand side expressions using `<Diff>`, `<PartialDiff>` and other elements described below
  - `<IndependentVariable>` – differentiation variable
  - `<InitialCondition>` – initial condition specification for DEs
    - `<InitialValue>`
    - `<InitialTime>`
  - `<BoundaryCondition type="">` – boundary condition specification for DEs
    - `type` (optional) attribute with allowed values: *Cauchy*, *Dirichlet*, *Neumann*, *Robin*, *mixed*
    - `<ConditionVariable>` – independent variable for which the condition is specified
    - `<BoundaryValue>` – the value of the independent variable
    - `<AssignStatement>` – boundary condition expression
  - `<History>` – for DDEs
  - `type` (optional) attribute with allowed values
    - *dde*

[1] `https://www.w3.org/TR/MathML3/chapter4.html#id.4.4.4`

    - *ode*, *odeStiff*, *odeLinear*
    - *pde*, *pdeElliptic*, *pdeHyperbolic*, *pdeParabolic*, *pdeMixed*

- differentiation operator – `<Diff>`
  - `<Degree>` – (optional) degree of the operator, must be positive. Defaults to 1 if not specified otherwise.
  - `<DiffVariable>` – (optional, only one allowed) differentiation variable and its order. Defaults to `<IndependentVariable>` if not specified otherwise.
  - `<DiffOpArgument>` – argument (can be any expression) of the differentiation operator.
- partial differentiation operator – `<PartialDiff>`
  - `<Degree>` – (optional) total degree of the operator. Must be positive and equals the sum of all partial differentiation variable degrees. Can be omitted if partial differentiation is specified with respect to one variable only, e.g. $\partial^2 u/\partial x^2$.
  - `<DiffVariable>` – (multiple allowed) differentiation variables, in their order, with their corresponding
    - `<Degree>` – degree of partial differentiation variable
  - `<DiffOpArgument>` – argument (can be any expression) of the differentiation operator.
- vector differential calculus operators – `<VectorCalcOp op="">`, expressed in terms of ∇, the nabla/del operator
  - attribute `op` taking values
    - `curl` $\equiv \nabla\times$ – curl operator
    - `divergence` $\equiv \nabla\cdot$ – divergence operator
    - `gradient` $\equiv \nabla$ – gradient operator
    - `laplacian` $\equiv \nabla^2$ – laplace operator
  - `<DiffVariable>` – differentiation variables, scalar or vector
  - `<DiffOpArgument>` – argument (can be any expression) of the differentiation operator
- integration – `<Integral>`
  - `<LowLimit>`, `<UpLimit>` – lower and upper integration limit
  - `<IntegrationVariable>` – integration variable
  - `<IntegralArgument>` – integration operator argument

N-ary operator	Element name	Argument	Return type
Integration	`<Integral>`	`x`	`y`
Differentiation	`<Diff>`	`x`	`y`
Partial differentiation	`<PartialDiff>`	`x`	`y`
Curl	`<VectorCalcOp op="curl">`	`X`	`Y`
Divergence	`<VectorCalcOp op="divergence>`	`X`	`y`
Gradient	`<VectorCalcOp op="gradient>`	`x`	`Y`
Laplace operator	`<VectorCalcOp op="laplacian>`	`x`	`y`

Table 3.1: Calculus and vector differential operators. `x` and `y` are real scalars, `X` and `Y` are real vectors.

Detailed examples of ODEs and PDEs with different boundary and initial conditions will follow next.

3.4 Initial examples

3.4.1 Encoding differential equations

ODE template

The following example is for illustration only, any expression on the LHS/RHS is permitted.

$$\frac{dA}{dt} = \text{RHS expression} \quad \text{with} \quad \text{Initial Condition} : Y(t=t0) = Y0$$

```
<!-- dA/dt = RHS; Initial Condition: A(t=...) = ...-->
<ct:Variable symbId="A"/>
<ct:DE type="ode">
    <ct:AssignStatement op="eq">
        <!-- LHS expression -->
        <!-- RHS expression -->
    </ct:AssignStatement>
    <ct:InitialCondition>
        <ct:InitialValue>
            <!-- ... -->
        </ct:InitialValue>
        <ct:InitialTime>
            <!-- ... -->
        </ct:InitialTime>
    </ct:InitialCondition>
</ct:DE>
```

Note, that expressions on the LHS are allowed as well, see example 2 in section 3.6. Also permitted are differentials on the RHS, see example 3 in section 3.6.

PDE template

The following example is for illustration only, any expression on the LHS/RHS is permitted, both for the PDE and the conditions.

$$\frac{\partial^3 A}{\partial t \partial x^2} = \text{RHS expression}$$

$$\begin{aligned} \text{IC}, t = t0: &\quad Y0 \\ \text{BC}, x = x_{min}: &\quad \text{LHS expression} = \text{RHS expression} \\ x = x_{max}: &\quad \text{LHS expression} = \text{RHS expression} \end{aligned}$$

```
<!-- PDE: LHS = RHS -->
<!-- ID: A(t=t0)=Y0, BC: x=x_min: LHS = RHS; x=x_max: LHS = RHS -->
<ct:Variable symbId="A"/>
<ct:DE type="pde">

    <ct:AssignStatement op="eq">
        <!-- LHS of the PDE -->
        <!-- RHS of the PDE -->
    </ct:AssignStatement>

    <ct:InitialCondition>
        <ct:ConditionVariable>
            <ct:SymbRef symbIdRef="t"/>
        </ct:ConditionVariable>
        <ct:InitialValue>
            <!-- ... -->
        </ct:InitialValue>
        <ct:InitialTime>
            <!-- ... -->
        </ct:InitialTime>
    </ct:InitialCondition>
```

```
            <ct:BoundaryCondition type="Neumann">
                <ct:ConditionVariable>
                    <ct:SymbRef symbIdRef="x"/>
                </ct:ConditionVariable>
                <ct:BoundaryValue>
                    <ct:Assign>
                        <!-- x_min -->
                    </ct:Assign>
                </ct:BoundaryValue>
                <ct:AssignStatement op="eq">
                    <!-- ... -->
                </ct:AssignStatement>
            </ct:BoundaryCondition>

        </ct:DE>
```

3.4.2 Encoding new operators

In this section we collected isolated examples of the encoding of calculus and vector differential operators to illustrate their use.

Integral

$$\int_{x_1}^{x_2} A \, dx$$

```
            <math:Integral>
                <math:LowLimit>
                    <ct:SymbRef symbIdRef="x1"/>
                </math:LowLimit>
                <math:UpLimit>
                    <ct:SymbRef symbIdRef="x2"/>
                </math:UpLimit>
                <math:IntegrationVariable>
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="x"/>
                    </ct:Assign>
                </math:IntegrationVariable>
                <math:IntegralArgument>
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="A"/>
                    </ct:Assign>
                </math:IntegralArgument>
            </math:Integral>
```

Diff

$$\frac{dA}{dt}$$

```
            <math:Diff>
                <math:DiffVariable>
                    <ct:SymbRef symbIdRef="t"/>
                </math:DiffVariable>
                <math:DiffOpArgument>
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="A"/>
                    </ct:Assign>
                </math:DiffOpArgument>
            </math:Diff>
```

PartialDiff

$$\frac{\partial^3 A}{\partial t \partial x^2}$$

```
<math:PartialDiff>
    <math:Degree>
        <ct:Assign>
            <ct:Real>3</ct:Real>
        </ct:Assign>
    </math:Degree>
    <math:DiffVariable>
        <ct:SymbRef symbIdRef="t"/>
        <math:Degree>
            <ct:Assign>
                <ct:Real>1</ct:Real>
            </ct:Assign>
        </math:Degree>
    </math:DiffVariable>
    <math:DiffVariable>
        <ct:SymbRef symbIdRef="x"/>
        <math:Degree>
            <ct:Assign>
                <ct:Real>2</ct:Real>
            </ct:Assign>
        </math:Degree>
    </math:DiffVariable>
    <math:DiffOpArgument>
        <ct:Assign>
            <ct:SymbRef symbIdRef="A"/>
        </ct:Assign>
    </math:DiffOpArgument>
</math:PartialDiff>
```

divergence

$$\nabla \cdot F$$

```
<math:VectorCalcOp op="divergence">
    <math:DiffVariables>
        <ct:Assign>
            <ct:Vector>
                <ct:VectorElements>
                    <ct:SymbRef symbIdRef="x"/>
                    <ct:SymbRef symbIdRef="y"/>
                    <ct:SymbRef symbIdRef="z"/>
                </ct:VectorElements>
            </ct:Vector>
        </ct:Assign>
    </math:DiffVariables>
    <math:DiffOpArgument>
        <ct:Assign>
            <ct:SymbRef symbIdRef="F"/>
        </ct:Assign>
    </math:DiffOpArgument>
</math:VectorCalcOp>
```

gradient

$$\nabla f$$

```
<math:VectorCalcOp op="gradient">
    <math:DiffVariables>
        <ct:Assign>
            <ct:SymbRef symbIdRef="x"/>
```

```
            </ct:Assign>
        </math:DiffVariables>
        <math:DiffOpArgument>
            <ct:Assign>
                <ct:SymbRef symbIdRef="f"/>
            </ct:Assign>
        </math:DiffOpArgument>
    </math:VectorCalcOp>
```

laplacian

$$\nabla^2 f$$

```
    <math:VectorCalcOp op="laplacian">
        <math:DiffVariables>
            <ct:Assign>
                <ct:SymbRef symbIdRef="x"/>
                <ct:SymbRef symbIdRef="y"/>
            </ct:Assign>
        </math:DiffVariables>
        <math:DiffOpArgument>
            <ct:Assign>
                <ct:SymbRef symbIdRef="f"/>
            </ct:Assign>
        </math:DiffOpArgument>
    </math:VectorCalcOp>
```

3.5 Integration example

A basic example, e.g.

$$\int_a^b f(x)\, dx = 1$$

reads in PharmML

```
<ct:AssignStatement op="eq">
    <math:Integral>
        <math:LowLimit>
            <ct:Assign>
                <ct:SymbRef symbIdRef="a"/>
            </ct:Assign>
        </math:LowLimit>
        <math:UpLimit>
            <ct:Assign>
                <ct:SymbRef symbIdRef="b"/>
            </ct:Assign>
        </math:UpLimit>
        <math:IntegrationVariable>
            <ct:Assign>
                <ct:SymbRef symbIdRef="x"/>
            </ct:Assign>
        </math:IntegrationVariable>
        <math:IntegralArgument>
            <ct:Assign>
                <ct:SymbRef symbIdRef="f"/>
            </ct:Assign>
        </math:IntegralArgument>
    </math:Integral>
    <ct:Real>1</ct:Real>
</ct:AssignStatement>
```

See a real-life example in section 4.2.

3.6 ODE examples

3.6.1 Basic ODE

$$\frac{dQ_{ing}}{dt} = Q_{ing_{rate}} \quad \text{with} \quad Q_{ing}(t=0) = Q0$$

```
<!-- dt(Q_ing) = Q_ing_rate with Q_ing(t=0) = Q_0-->
<ct:Variable symbId="Q_ing"/>
<ct:DE type="ode">
    <ct:AssignStatement op="eq">
        <math:Diff>
            <math:Degree>
                <ct:Assign>
                    <ct:Int>1</ct:Int>
                </ct:Assign>
            </math:Degree>
            <math:DiffVariable>
                <ct:SymbRef symbIdRef="t"/>
            </math:DiffVariable>
            <math:DiffOpArgument>
                <ct:Assign>
                    <ct:SymbRef symbIdRef="Q_ing"/>
                </ct:Assign>
            </math:DiffOpArgument>
        </math:Diff>
        <ct:SymbRef symbIdRef="Q_ing_rate"/>
    </ct:AssignStatement>
    <ct:InitialCondition>
        <ct:InitialValue>
            <ct:Assign>
                <ct:SymbRef symbIdRef="Q_0"/>
            </ct:Assign>
        </ct:InitialValue>
        <ct:InitialTime>
            <ct:Assign>
                <ct:Real>0</ct:Real>
            </ct:Assign>
        </ct:InitialTime>
    </ct:InitialCondition>
</ct:DE>
```

3.6.2 ODE with an expressions on LHS

Based on [Mager, 2006].

$$\frac{d(V_T C_T)}{dt} = Q_T\left(C_A - \frac{C_T}{K_{PT}}\right)$$

```
<ct:Variable symbId="C_T"/>
<ct:DE type="ode">
    <ct:AssignStatement op="eq">
        <math:Diff>
            <math:DiffVariable>
                <ct:SymbRef symbIdRef="t"/>
            </math:DiffVariable>
            <math:DiffOpArgument>
                <ct:Assign>
                    <math:Binop op="times">
                        <ct:SymbRef symbIdRef="V_T"/>
                        <ct:SymbRef symbIdRef="C_T"/>
                    </math:Binop>
                </ct:Assign>
            </math:DiffOpArgument>
        </math:Diff>
        <math:Binop op="times">
            <ct:SymbRef symbIdRef="Q_T"/>
```

```
                    <math:Binop op="minus">
                        <ct:SymbRef symbIdRef="C_A"/>
                        <math:Binop op="divide">
                            <ct:SymbRef symbIdRef="C_T"/>
                            <ct:SymbRef symbIdRef="K_PT"/>
                        </math:Binop>
                    </math:Binop>
                </math:Binop>
            </ct:AssignStatement>
        </ct:DE>
```

or

$$V_T \frac{dC_T}{dt} = Q_T\left(C_A - \frac{C_T}{K_{PT}}\right)$$

```
        <ct:DE type="ode">
            <ct:AssignStatement op="eq">
                <math:Binop op="times">
                    <ct:SymbRef symbIdRef="V_T"/>
                    <math:Diff>
                        <math:DiffVariable>
                            <ct:SymbRef symbIdRef="t"/>
                        </math:DiffVariable>
                        <math:DiffOpArgument>
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="C_T"/>
                            </ct:Assign>
                        </math:DiffOpArgument>
                    </math:Diff>
                </math:Binop>
                <math:Binop op="times">
                    <ct:SymbRef symbIdRef="Q_T"/>
                    <math:Binop op="minus">
                        <ct:SymbRef symbIdRef="C_A"/>
                        <math:Binop op="divide">
                            <ct:SymbRef symbIdRef="C_T"/>
                            <ct:SymbRef symbIdRef="K_PT"/>
                        </math:Binop>
                    </math:Binop>
                </math:Binop>
            </ct:AssignStatement>
        </ct:DE>
```

3.6.3 ODE with differentials on RHS

$$\begin{aligned} &\dots \\ \frac{dQ_{urine}}{dt} &= Ke_{renal} C_{kidney} \\ \frac{dQ_{kidney}}{dt} &= F_{kidney}(C_{art} - C_{kidney_v}) - \frac{dQ_{urine}}{dt} \\ &\dots \end{aligned}$$

```
        <!-- dt(Q_kidney) = F_kidney * (C_art - C_kidney_v) - dt(Q_urine) -->
        <ct:Variable symbId="Q_kidney"/>
        <ct:DE>
            <ct:AssignStatement op="and">
                <math:Diff>
                    <math:DiffOpArgument>
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="Q_kidney"/>
                        </ct:Assign>
                    </math:DiffOpArgument>
                </math:Diff>
                <math:Binop op="minus">
                    <math:Binop op="times">
                        <ct:SymbRef symbIdRef="F_kidney"/>
```

```
                    <math:Binop op="minus">
                        <ct:SymbRef symbIdRef="C_art"/>
                        <ct:SymbRef symbIdRef="C_kidney_v"/>
                    </math:Binop>
                </math:Binop>
                <math:Diff>
                    <!-- Degree and DiffVariable default to '1' and 't', respectively -->
                    <math:DiffOpArgument>
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="Q_urine"/>
                        </ct:Assign>
                    </math:DiffOpArgument>
                </math:Diff>
            </math:Binop>
        </ct:AssignStatement>
    </ct:DE>
```

3.7 PDE examples

3.7.1 Advection Equation

Source: `http://www.physiome.org/jsim/docs/MML_PDE.html#adv`

Model definition

$$\frac{\partial C(x,t)}{\partial t} = -U\frac{\partial C(x,t)}{\partial x}$$

$$IC, t = t_{min}: \quad C = \begin{cases} Cin & \text{for} \quad x = xmin \\ C0 & \text{else} \end{cases}$$

$$BC, x = x_{min}: \quad -UC = -UCin, \quad (Dirichlet)$$

$$x = x_{max}: \quad \frac{\partial C}{\partial x} = 0, \quad (Neumann)$$

$$x = x_{max}: \quad C = Cout$$

JSIM notation

```
// Initial Condition
when(t=t.min) {C(x,t)=if(x=x.min) Cin else C0;}

// Boundary Conditions
when (x=x.min) { -U*C = -U*Cin;} // Left Hand BC
when (x=x.max) { C:x = 0; Cout = C;} // Right Hand BC

// Partial Differential Equation
C:t = -U*C:x ;
```

NOTE

Cin and $C0$ are forcing functions, called *extern* in JSim and in PharmML are declared as *regressors*. It is unclear from the JSim code where $Cout$ is coming from. The numerical settings, $Ngridx, t.delta$, are not implemented, they belong to the task description in the `<ModellingSteps>` and are skipped here.

PharmML implementation

Parameter model

```
<ParameterModel blkId="pm1">

   <!-- realDomain t sec; t.min=0; t.max=15; t.delta=0.1;
   real L = 0.1 cm, // Length of tube
```

```
        Ngridx = 51; // Number of grid points spatially
        realDomain x cm; x.min=0; x.max=L; x.ct=Ngridx;-->
        <Parameter symbId="tmin">
            <ct:Assign>
                <ct:Real>0</ct:Real>
            </ct:Assign>
        </Parameter>
        <Parameter symbId="tmax">
            <ct:Assign>
                <ct:Real>15</ct:Real>
            </ct:Assign>
        </Parameter>
        <Parameter symbId="L">
            <ct:Assign>
                <ct:Real>0.1</ct:Real>
            </ct:Assign>
        </Parameter>
        <Parameter symbId="xmin">
            <ct:Assign>
                <ct:Real>0</ct:Real>
            </ct:Assign>
        </Parameter>
        <Parameter symbId="xmax">
            <ct:Assign>
                <ct:SymbRef symbIdRef="L"/>
            </ct:Assign>
        </Parameter>

    </ParameterModel>
```

Differential equation

```
    <StructuralModel blkId="sm1">

        <ct:Variable symbId="C"/>

        <ct:Variable regressor="yes" symbId="Cin"/>
        <ct:Variable regressor="yes" symbId="C0"/>

        <ct:DE type="pdeHyperbolic">
            <ct:AssignStatement op="eq">
                <math:PartialDiff>
                    <math:DiffVariable>
                        <ct:SymbRef symbIdRef="t"/>
                    </math:DiffVariable>
                    <math:DiffOpArgument>
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="C"/>
                        </ct:Assign>
                    </math:DiffOpArgument>
                </math:PartialDiff>
                <math:Binop op="times">
                    <math:Uniop op="minus">
                        <ct:SymbRef symbIdRef="U"/>
                    </math:Uniop>
                    <math:PartialDiff>
                        <math:DiffVariable>
                            <ct:SymbRef symbIdRef="x"/>
                        </math:DiffVariable>
                        <math:DiffOpArgument>
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="C"/>
                            </ct:Assign>
                        </math:DiffOpArgument>
                    </math:PartialDiff>
                </math:Binop>
            </ct:AssignStatement>
```

Initial conditions

```
            <ct:InitialCondition>
                <ct:InitialValue>
```

```
                <ct:Assign>
                    <math:Piecewise>
                        <math:Piece>
                            <ct:SymbRef symbIdRef="Cin"/>
                            <math:Condition>
                                <math:LogicBinop op="eq">
                                    <ct:SymbRef symbIdRef="x"/>
                                    <ct:SymbRef blkIdRef="pm1" symbIdRef="xmin"/>
                                </math:LogicBinop>
                            </math:Condition>
                        </math:Piece>
                        <math:Piece>
                            <ct:SymbRef symbIdRef="C0"/> <!-- regressor, 'extern' in JSim -->
                            <math:Condition>
                                <math:Otherwise/>
                            </math:Condition>
                        </math:Piece>
                    </math:Piecewise>
                </ct:Assign>
            </ct:InitialValue>
        </ct:InitialCondition>
```

Dirichlet boundary conditions

```
        <ct:BoundaryCondition type="Dirichlet">
            <ct:ConditionVariable>
                <ct:SymbRef symbIdRef="x"/>
            </ct:ConditionVariable>
            <ct:BoundaryValue>
                <ct:Assign>
                    <ct:SymbRef blkIdRef="pm1" symbIdRef="xmin"/>
                </ct:Assign>
            </ct:BoundaryValue>
            <ct:AssignStatement op="eq">
                <math:Binop op="times">
                    <math:Uniop op="minus">
                        <ct:SymbRef symbIdRef="U"/>
                    </math:Uniop>
                    <ct:SymbRef symbIdRef="C"/>
                </math:Binop>
                <math:Binop op="times">
                    <math:Uniop op="minus">
                        <ct:SymbRef symbIdRef="U"/>
                    </math:Uniop>
                    <ct:SymbRef symbIdRef="Cin"/> <!-- regressor, 'extern' in JSim -->
                </math:Binop>
            </ct:AssignStatement>
        </ct:BoundaryCondition>

        <ct:BoundaryCondition type="Neumann">
            <ct:ConditionVariable>
                <ct:SymbRef symbIdRef="x"/>
            </ct:ConditionVariable>
            <ct:BoundaryValue>
                <ct:Assign>
                    <ct:SymbRef blkIdRef="pm1" symbIdRef="xmax"/>
                </ct:Assign>
            </ct:BoundaryValue>
            <ct:AssignStatement op="eq">
                <math:PartialDiff>
                    <math:DiffVariable>
                        <ct:SymbRef symbIdRef="x"/>
                    </math:DiffVariable>
                    <math:DiffOpArgument>
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="C"/>
                        </ct:Assign>
                    </math:DiffOpArgument>
                </math:PartialDiff>
                <ct:Real>0</ct:Real>
            </ct:AssignStatement>
        </ct:BoundaryCondition>
```

```
            <ct:BoundaryCondition>
                <ct:ConditionVariable>
                    <ct:SymbRef symbIdRef="x"/>
                </ct:ConditionVariable>
                <ct:BoundaryValue>
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="xmax"/>
                    </ct:Assign>
                </ct:BoundaryValue>
                <ct:AssignStatement op="eq">
                    <ct:SymbRef symbIdRef="C"/>
                    <ct:SymbRef symbIdRef="Cout"/>
                </ct:AssignStatement>
            </ct:BoundaryCondition>

        </ct:DE>
    </StructuralModel>
```

3.7.2 Diffusion Equation

Source: `http://www.physiome.org/jsim/docs/MML_PDE.html#diff`

Model definition

$$\frac{\partial C(x,t)}{\partial t} = D\frac{\partial^2 C(x,t)}{\partial x^2}$$

$$IC, t = t_{min}: \quad C = C0$$

$$BC, x = x_{min}: \quad D\frac{\partial C}{\partial x} = 0, \quad (Neumann)$$

$$x = x_{max}: \quad D\frac{\partial C}{\partial x} = 0, \quad (Neumann)$$

JSIM notation

```
    // Initial Condition
    when(t=t.min) {C(x,t) = C0;}

    // Boundary Conditions
    when (x=x.min) { D*C:x = 0;} // Left Hand BC
    when (x=x.max) { D*C:x = 0; } // Right Hand BC

    // Partial Differential Equation
    C:t = D*C:x:x;
```

Parameter model

```
    <ParameterModel blkId="pm1">
        <!-- realDomain t sec; t.min=0; t.max=15; t.delta=0.1;
            real L = 0.1 cm, // Length of tube
            Ngridx = 51; // Number of grid divisions
            realDomain x cm; x.min=0; x.max=L; x.ct=Ngridx;
            real D = 0.00005 cm^2/sec; // Diffusion coefficient -->
        <Parameter symbId="tmin">
            <ct:Assign>
                <ct:Real>0</ct:Real>
            </ct:Assign>
        </Parameter>
        <Parameter symbId="tmax">
            <ct:Assign>
                <ct:Real>15</ct:Real>
            </ct:Assign>
        </Parameter>
        <Parameter symbId="L">
            <ct:Assign>
```

```
                    <ct:Real>0.1</ct:Real>
                </ct:Assign>
            </Parameter>
            <Parameter symbId="xmin">
                <ct:Assign>
                    <ct:Real>0</ct:Real>
                </ct:Assign>
            </Parameter>
            <Parameter symbId="xmax">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="L"/>
                </ct:Assign>
            </Parameter>
            <Parameter symbId="D">
                <ct:Assign>
                    <ct:Real>0.00005</ct:Real>
                </ct:Assign>
            </Parameter>
        </ParameterModel>
```

Differential equation

```
        <StructuralModel blkId="sm1">

<!-- real C(x,t) mM; // Concentration of substance
            extern real C0(x) mM; // Initial Distribution of C-->
            <ct:Variable symbId="C"/>
            <ct:Variable regressor="yes" symbId="C0"/>

<!-- // Partial Differential Equation
            C:t = D*C:x:x;-->
            <ct:DE type="pdeParabolic">
                <ct:AssignStatement op="eq">
                    <math:PartialDiff>
                        <math:DiffVariable>
                            <ct:SymbRef symbIdRef="t"/>
                        </math:DiffVariable>
                        <math:DiffOpArgument>
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="C"/>
                            </ct:Assign>
                        </math:DiffOpArgument>
                    </math:PartialDiff>
                    <math:Binop op="times">
                        <ct:SymbRef symbIdRef="D"/>
                        <math:PartialDiff>
                            <math:DiffVariable>
                                <ct:SymbRef symbIdRef="x"/>
                                <math:Degree>
                                    <ct:Assign>
                                        <ct:Int>2</ct:Int>
                                    </ct:Assign>
                                </math:Degree>
                            </math:DiffVariable>
                            <math:DiffOpArgument>
                                <ct:Assign>
                                    <ct:SymbRef symbIdRef="C"/>
                                </ct:Assign>
                            </math:DiffOpArgument>
                        </math:PartialDiff>
                    </math:Binop>
                </ct:AssignStatement>
```

Initial conditions

```
<!-- // Initial Condition
                when(t=t.min) {C(x,t) = C0;}-->
                <ct:InitialCondition>
                    <ct:ConditionVariable>
                        <ct:SymbRef symbIdRef="t"/>
                    </ct:ConditionVariable>
                    <ct:InitialValue>
```

```
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="C0"/>
                    </ct:Assign>
                </ct:InitialValue>
                <ct:InitialTime>
                    <ct:Assign>
                        <ct:SymbRef blkIdRef="pm1" symbIdRef="tmin"/>
                    </ct:Assign>
                </ct:InitialTime>
            </ct:InitialCondition>
```

Neumann boundary conditions

```
<!-- // Boundary Conditions
            when (x=x.min) { D*C:x = 0;} // Left Hand BC
            when (x=x.max) { D*C:x = 0; } // Right Hand BC-->
            <ct:BoundaryCondition type="Neumann">
                <ct:ConditionVariable>
                    <ct:SymbRef symbIdRef="x"/>
                </ct:ConditionVariable>
                <ct:BoundaryValue>
                    <ct:Assign>
                        <ct:SymbRef blkIdRef="pm1" symbIdRef="xmin"/>
                    </ct:Assign>
                </ct:BoundaryValue>
                <ct:AssignStatement op="eq">
                    <math:Binop op="times">
                        <ct:SymbRef symbIdRef="D"/>
                        <math:PartialDiff>
                            <math:DiffVariable>
                                <ct:SymbRef symbIdRef="x"/>
                            </math:DiffVariable>
                            <math:DiffOpArgument>
                                <ct:Assign>
                                    <ct:SymbRef symbIdRef="C"/>
                                </ct:Assign>
                            </math:DiffOpArgument>
                        </math:PartialDiff>
                    </math:Binop>
                    <ct:Real>0</ct:Real>
                </ct:AssignStatement>
            </ct:BoundaryCondition>

            <ct:BoundaryCondition type="Neumann">
                <ct:ConditionVariable>
                    <ct:SymbRef symbIdRef="x"/>
                </ct:ConditionVariable>
                <ct:BoundaryValue>
                    <ct:Assign>
                        <ct:SymbRef blkIdRef="pm1" symbIdRef="xmax"/>
                    </ct:Assign>
                </ct:BoundaryValue>
                <ct:AssignStatement op="eq">
                    <math:Binop op="times">
                        <ct:SymbRef symbIdRef="D"/>
                        <math:PartialDiff>
                            <math:DiffVariable>
                                <ct:SymbRef symbIdRef="x"/>
                            </math:DiffVariable>
                            <math:DiffOpArgument>
                                <ct:Assign>
                                    <ct:SymbRef symbIdRef="C"/>
                                </ct:Assign>
                            </math:DiffOpArgument>
                        </math:PartialDiff>
                    </math:Binop>
                    <ct:Real>0</ct:Real>
                </ct:AssignStatement>
            </ct:BoundaryCondition>
        </ct:DE>
```

3.7.3 Combined Advection/Diffusion Equation

Source: http://www.physiome.org/jsim/docs/MML_PDE.html#advdiff

Model definition

$$\frac{\partial C(x,t)}{\partial t} = -U\frac{\partial C(x,t)}{\partial x} + D\frac{\partial^2 C(x,t)}{\partial x^2}$$

$$IC, t = t_{min}: \quad C = \begin{cases} Cin & \text{for} \quad x = xmin \\ C0 & \text{else} \end{cases}$$

$$BC, x = x_{min}: \quad -U\,C + D\frac{\partial C}{\partial x} = -U\,Cin,$$

$$x = x_{max}: \quad D\,\frac{\partial C}{\partial x} = 0, \quad (Neumann)$$

$$x = x_{max}: \quad C = Cout$$

JSIM notation

```
// Initial Condition
when(t=t.min) {C(x,t)=if(x=x.min) Cin else C0;}

// Boundary Conditions
when (x=x.min) { -U*C + D*C:x = -U*Cin; } // Left Hand Total Flux BC
when (x=x.max) { D*C:x=0; Cout=C;} // Right Hand Neumann BC

// Partial Differential Equation
C:t = -U*C:x + D*C:x:x;
```

Regarding $Cin(t)$ and $C0(x)$, see note in section 3.7.1.

PharmML implementation

Skipped here as similar the that in section 3.7.1 about the advection equation.

Differential equation

```
      <StructuralModel blkId="sm1">

<!-- extern real Cin(t) mM; // Inflow Concentration
          extern real C0(x) mM; // Initial Distribution of C-->
          <ct:Variable symbId="C"/>
          <ct:Variable regressor="yes" symbId="Cin"/>
          <ct:Variable regressor="yes" symbId="C0"/>

<!-- // Partial Differential Equation
          C:t = -U*C:x + D*C:x:x;-->
          <ct:DE type="pdeParabolic">
              <ct:AssignStatement op="eq">
                  <math:PartialDiff>
                      <math:DiffVariable>
                          <ct:SymbRef symbIdRef="t"/>
                      </math:DiffVariable>
                      <math:DiffOpArgument>
                          <ct:Assign>
                              <ct:SymbRef symbIdRef="C"/>
                          </ct:Assign>
                      </math:DiffOpArgument>
                  </math:PartialDiff>
                  <math:Binop op="plus">
                      <math:Binop op="times">
                          <math:Uniop op="minus">
                              <ct:SymbRef symbIdRef="U"/>
                          </math:Uniop>
```

```
                        <math:PartialDiff>
                            <math:DiffVariable>
                                <ct:SymbRef symbIdRef="x"/>
                            </math:DiffVariable>
                            <math:DiffOpArgument>
                                <ct:Assign>
                                    <ct:SymbRef symbIdRef="C"/>
                                </ct:Assign>
                            </math:DiffOpArgument>
                        </math:PartialDiff>
                    </math:Binop>
                    <math:Binop op="times">
                        <ct:SymbRef symbIdRef="D"/>
                        <math:PartialDiff>
                            <math:DiffVariable>
                                <ct:SymbRef symbIdRef="x"/>
                                <math:Degree>
                                    <ct:Assign>
                                        <ct:Int>2</ct:Int>
                                    </ct:Assign>
                                </math:Degree>
                            </math:DiffVariable>
                            <math:DiffOpArgument>
                                <ct:Assign>
                                    <ct:SymbRef symbIdRef="C"/>
                                </ct:Assign>
                            </math:DiffOpArgument>
                        </math:PartialDiff>
                    </math:Binop>
                </math:Binop>
            </ct:AssignStatement>
```

Initial conditions

Skipped here as identical the that in section 3.7.1 about the advection equation.

Boundary conditions

```
            <!-- -U*C + D*C:x = -U*Cin -->
            <ct:BoundaryCondition type="Neumann">
                <ct:ConditionVariable>
                    <ct:SymbRef symbIdRef="x"/>
                </ct:ConditionVariable>
                <ct:BoundaryValue>
                    <ct:Assign>
                        <ct:SymbRef blkIdRef="pm1" symbIdRef="xmin"/>
                    </ct:Assign>
                </ct:BoundaryValue>
                <ct:AssignStatement op="eq">
                    <math:Binop op="plus">
                        <math:Binop op="times">
                            <math:Uniop op="minus">
                                <ct:SymbRef symbIdRef="U"/>
                            </math:Uniop>
                            <ct:SymbRef symbIdRef="C"/>
                        </math:Binop>
                        <math:Binop op="times">
                            <ct:SymbRef symbIdRef="D"/>
                            <math:PartialDiff>
                                <math:DiffVariable>
                                    <ct:SymbRef symbIdRef="x"/>
                                </math:DiffVariable>
                                <math:DiffOpArgument>
                                    <ct:Assign>
                                        <ct:SymbRef symbIdRef="C"/>
                                    </ct:Assign>
                                </math:DiffOpArgument>
                            </math:PartialDiff>
                        </math:Binop>
                    </math:Binop>
                    <math:Binop op="times">
                        <math:Uniop op="minus">
```

```
                        <ct:SymbRef symbIdRef="U"/>
                    </math:Uniop>
                    <ct:SymbRef symbIdRef="Cin"/>
                </math:Binop>
            </ct:AssignStatement>
        </ct:BoundaryCondition>

        <!-- D*C:x=0 -->
        <ct:BoundaryCondition type="Neumann">
            <ct:ConditionVariable>
                <ct:SymbRef symbIdRef="x"/>
            </ct:ConditionVariable>
            <ct:BoundaryValue>
                <ct:Assign>
                    <ct:SymbRef blkIdRef="pm1" symbIdRef="xmax"/>
                </ct:Assign>
            </ct:BoundaryValue>
            <ct:AssignStatement op="eq">
                <math:Binop op="times">
                    <ct:SymbRef symbIdRef="D"/>
                    <math:PartialDiff>
                        <math:DiffVariable>
                            <ct:SymbRef symbIdRef="x"/>
                        </math:DiffVariable>
                        <math:DiffOpArgument>
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="C"/>
                            </ct:Assign>
                        </math:DiffOpArgument>
                    </math:PartialDiff>
                </math:Binop>
                <ct:Real>0</ct:Real>
            </ct:AssignStatement>
        </ct:BoundaryCondition>

        <!-- Cout=C -->
        <ct:BoundaryCondition type="Neumann">
            <ct:ConditionVariable>
                <ct:SymbRef symbIdRef="x"/>
            </ct:ConditionVariable>
            <ct:BoundaryValue>
                <ct:Assign>
                    <ct:SymbRef blkIdRef="pm1" symbIdRef="xmax"/>
                </ct:Assign>
            </ct:BoundaryValue>
            <ct:AssignStatement op="eq">
                <ct:SymbRef symbIdRef="Cout"/>
                <ct:SymbRef symbIdRef="C"/>
            </ct:AssignStatement>
        </ct:BoundaryCondition>
    </ct:DE>
```

3.7.4 Diffusion Equation – Ghaffarizadeh 2016

Source: [Ghaffarizadeh et al., 2016], Appendix example.

Model definition

$$\frac{\partial \rho(x,t)}{\partial t} = D\frac{\partial^2 \rho(x,t)}{\partial x^2}$$

$$IC, t = t_{min}: \quad \rho(x,t) = 1 + \cos\left(\frac{\pi}{L_0}x\right)$$

$$BC, x = x_{min}: \quad D\frac{\partial \rho}{\partial x} = 0, \quad (Neumann)$$

$$x = x_{max}: \quad D\frac{\partial \rho}{\partial x} = 0, \quad (Neumann)$$

with $t_{min} = 0$, $x_{min} = -L0$ and $x_{max} = L0$.

PharmML implementation

Parameter model

```
<ParameterModel blkId="pm1">

    <Parameter symbId="D">
        <ct:Assign>
            <ct:Real>100000</ct:Real>
        </ct:Assign>
    </Parameter>

    <!-- tmin=0; L0 = 500; xmin=-L0; xmax=L0 -->
    <Parameter symbId="tmin">
        <ct:Assign>
            <ct:Real>0</ct:Real>
        </ct:Assign>
    </Parameter>
    <Parameter symbId="L0">
        <ct:Assign>
            <ct:Real>500</ct:Real>
        </ct:Assign>
    </Parameter>
    <Parameter symbId="xmin">
        <ct:Assign>
            <math:Uniop op="minus">
                <ct:SymbRef symbIdRef="L0"/>
            </math:Uniop>
        </ct:Assign>
    </Parameter>
    <Parameter symbId="xmax">
        <ct:Assign>
            <ct:SymbRef symbIdRef="L0"/>
        </ct:Assign>
    </Parameter>
</ParameterModel>
```

Differential equation

```
    <!-- rho:t = D*rho:x:x;-->
    <ct:DE type="pdeParabolic">

        <ct:AssignStatement op="eq">
            <math:PartialDiff>
                <math:DiffVariable>
                    <ct:SymbRef symbIdRef="t"/>
                </math:DiffVariable>
                <math:DiffOpArgument>
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="rho"/>
                    </ct:Assign>
                </math:DiffOpArgument>
            </math:PartialDiff>
            <math:Binop op="times">
                <ct:SymbRef symbIdRef="D"/>
                <math:PartialDiff>
                    <math:DiffVariable>
                        <ct:SymbRef symbIdRef="x"/>
                        <math:Degree>
                            <ct:Assign>
                                <ct:Int>2</ct:Int>
                            </ct:Assign>
                        </math:Degree>
                    </math:DiffVariable>
                    <math:DiffOpArgument>
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="rho"/>
                        </ct:Assign>
                    </math:DiffOpArgument>
                </math:PartialDiff>
            </math:Binop>
```

```
        </ct:AssignStatement>
```

Initial condition

```
        <!-- t=tmin: rho = 1 + cos(...) -->
        <ct:InitialCondition>
            <ct:ConditionVariable>
                <ct:SymbRef symbIdRef="t"/>
            </ct:ConditionVariable>
            <ct:InitialValue>
                <ct:Assign>
                    <math:Binop op="plus">
                        <ct:Real>1</ct:Real>
                        <math:Uniop op="cos">
                            <math:Binop op="times">
                                <math:Binop op="divide">
                                    <math:Constant op="pi"/>
                                    <ct:SymbRef blkIdRef="pm1" symbIdRef="L0"/>
                                </math:Binop>
                                <ct:SymbRef symbIdRef="x"/>
                            </math:Binop>
                        </math:Uniop>
                    </math:Binop>
                </ct:Assign>
            </ct:InitialValue>
            <ct:InitialTime>
                <ct:Assign>
                    <ct:SymbRef symbIdRef="tmin"/>
                </ct:Assign>
            </ct:InitialTime>
        </ct:InitialCondition>
```

Boundary conditions

```
        <!--
        when (x=xmin) { rho:x = 0;}
        when (x=xmax) { rho:x = 0; } -->
        <ct:BoundaryCondition type="Neumann">
            <ct:ConditionVariable>
                <ct:SymbRef symbIdRef="x"/>
            </ct:ConditionVariable>
            <ct:BoundaryValue>
                <ct:Assign>
                    <ct:SymbRef blkIdRef="pm1" symbIdRef="xmin"/>
                </ct:Assign>
            </ct:BoundaryValue>
            <ct:AssignStatement op="eq">
                <math:PartialDiff>
                    <math:DiffVariable>
                        <ct:SymbRef symbIdRef="x"/>
                    </math:DiffVariable>
                    <math:DiffOpArgument>
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="rho"/>
                        </ct:Assign>
                    </math:DiffOpArgument>
                </math:PartialDiff>
                <ct:Real>0</ct:Real>
            </ct:AssignStatement>
        </ct:BoundaryCondition>

        <ct:BoundaryCondition type="Neumann">
            <ct:ConditionVariable>
                <ct:SymbRef symbIdRef="x"/>
            </ct:ConditionVariable>
            <ct:BoundaryValue>
                <ct:Assign>
                    <ct:SymbRef blkIdRef="pm1" symbIdRef="xmax"/>
                </ct:Assign>
            </ct:BoundaryValue>
            <ct:AssignStatement op="eq">
                <math:PartialDiff>
```

```
                    <math:DiffVariable>
                        <ct:SymbRef symbIdRef="x"/>
                    </math:DiffVariable>
                    <math:DiffOpArgument>
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="rho"/>
                        </ct:Assign>
                    </math:DiffOpArgument>
                </math:PartialDiff>
                <ct:Real>0</ct:Real>
            </ct:AssignStatement>
        </ct:BoundaryCondition>
```

3.7.5 Heat equation with different boundary conditions

Model definition

$$\frac{\partial u(x,t)}{\partial t} - \kappa \frac{\partial^2 u(x,t)}{\partial x^2} = 0$$

PharmML implementation

For simplicity the parameter model is skipped here and in the following examples.

Differential equation

```
<ct:DE>
    <ct:AssignStatement op="eq">
        <math:Binop op="minus">
            <math:PartialDiff>
                <math:DiffVariable>
                    <ct:SymbRef symbIdRef="t"/>
                </math:DiffVariable>
                <math:DiffOpArgument>
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="u"/>
                    </ct:Assign>
                </math:DiffOpArgument>
            </math:PartialDiff>
            <math:Binop op="times">
                <ct:SymbRef symbIdRef="kappa"/>
                <math:PartialDiff>
                    <math:DiffVariable>
                        <ct:SymbRef symbIdRef="x"/>
                        <math:Degree>
                            <ct:Assign>
                                <ct:Int>2</ct:Int>
                            </ct:Assign>
                        </math:Degree>
                    </math:DiffVariable>
                    <math:DiffOpArgument>
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="u"/>
                        </ct:Assign>
                    </math:DiffOpArgument>
                </math:PartialDiff>
            </math:Binop>
        </math:Binop>
        <ct:Real>0</ct:Real>
    </ct:AssignStatement>
```

Initial conditions

$$IC, t = 0: \quad u = f(x)$$

```
            <ct:InitialCondition>
                <ct:ConditionVariable>
                    <ct:SymbRef symbIdRef="t"/>
                </ct:ConditionVariable>
                <ct:InitialValue>
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="f"/>
                    </ct:Assign>
                </ct:InitialValue>
                <ct:InitialTime>
                    <ct:Assign>
                        <ct:Real>0</ct:Real>
                    </ct:Assign>
                </ct:InitialTime>
            </ct:InitialCondition>
```

Dirichlet boundary conditions

$$BC, x = 0: \quad u = T_0,$$
$$x = L: \quad u = T_L$$

```
            <ct:BoundaryCondition type="Dirichlet">
                <ct:ConditionVariable>
                    <ct:SymbRef symbIdRef="x"/>
                </ct:ConditionVariable>
                <ct:BoundaryValue>
                    <ct:Assign>
                        <ct:Real>0</ct:Real>
                    </ct:Assign>
                </ct:BoundaryValue>
                <ct:AssignStatement op="eq">
                    <ct:SymbRef symbIdRef="u"/>
                    <ct:SymbRef symbIdRef="T_0"/>
                </ct:AssignStatement>
            </ct:BoundaryCondition>
            <ct:BoundaryCondition type="Dirichlet">
                <ct:ConditionVariable>
                    <ct:SymbRef symbIdRef="x"/>
                </ct:ConditionVariable>
                <ct:BoundaryValue>
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="L"/>
                    </ct:Assign>
                </ct:BoundaryValue>
                <ct:AssignStatement op="eq">
                    <ct:SymbRef symbIdRef="u"/>
                    <ct:SymbRef symbIdRef="T_L"/>
                </ct:AssignStatement>
            </ct:BoundaryCondition>
        </ct:DE>
```

Neumann boundary conditions

$$BC, x = 0: \quad -\frac{\partial u}{\partial x} = \phi_0$$
$$x = L: \quad \frac{\partial u}{\partial x} = \phi_L$$

```
        <ct:DE>
            <!-- differential equation and ICs omitted -->
            <ct:BoundaryCondition type="Neumann">
                <ct:ConditionVariable>
                    <ct:SymbRef symbIdRef="x"/>
                </ct:ConditionVariable>
                <ct:BoundaryValue>
```

```
                <ct:Assign>
                    <ct:Real>0</ct:Real>
                </ct:Assign>
            </ct:BoundaryValue>
            <ct:AssignStatement op="eq">
                <math:Uniop op="minus">
                    <math:PartialDiff>
                        <math:DiffVariable>
                            <ct:SymbRef symbIdRef="x"/>
                        </math:DiffVariable>
                        <math:DiffOpArgument>
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="u"/>
                            </ct:Assign>
                        </math:DiffOpArgument>
                    </math:PartialDiff>
                </math:Uniop>
                <ct:SymbRef symbIdRef="phi_0"/>
            </ct:AssignStatement>
        </ct:BoundaryCondition>

        <ct:BoundaryCondition type="Neumann">
            <ct:ConditionVariable>
                <ct:SymbRef symbIdRef="x"/>
            </ct:ConditionVariable>
            <ct:BoundaryValue>
                <ct:Assign>
                    <ct:SymbRef symbIdRef="L"/>
                </ct:Assign>
            </ct:BoundaryValue>
            <ct:AssignStatement op="eq">
                <math:Uniop op="minus">
                    <math:PartialDiff>
                        <math:DiffVariable>
                            <ct:SymbRef symbIdRef="x"/>
                        </math:DiffVariable>
                        <math:DiffOpArgument>
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="u"/>
                            </ct:Assign>
                        </math:DiffOpArgument>
                    </math:PartialDiff>
                </math:Uniop>
                <ct:SymbRef symbIdRef="phi_L"/>
            </ct:AssignStatement>
        </ct:BoundaryCondition>
    </ct:DE>
```

Robin boundary values

$$BC, x = 0: \quad -\frac{\partial u}{\partial x} + h\, u = \psi_0$$
$$x = L: \quad \frac{\partial u}{\partial x} + h\, u = \psi_L$$

```
    <ct:DE>
        <!-- differential equation and ICs omitted -->

        <ct:BoundaryCondition type="Robin">
            <ct:ConditionVariable>
                <ct:SymbRef symbIdRef="x"/>
            </ct:ConditionVariable>
            <ct:BoundaryValue>
                <ct:Assign>
                    <ct:Real>0</ct:Real>
                </ct:Assign>
            </ct:BoundaryValue>
            <ct:AssignStatement op="eq">
                <math:Binop op="plus">
```

```
                    <math:Uniop op="minus">
                        <math:PartialDiff>
                            <math:DiffVariable>
                                <ct:SymbRef symbIdRef="x"/>
                            </math:DiffVariable>
                            <math:DiffOpArgument>
                                <ct:Assign>
                                    <ct:SymbRef symbIdRef="u"/>
                                </ct:Assign>
                            </math:DiffOpArgument>
                        </math:PartialDiff>
                    </math:Uniop>
                    <math:Binop op="times">
                        <ct:SymbRef symbIdRef="h"/>
                        <ct:SymbRef symbIdRef="u"/>
                    </math:Binop>
                </math:Binop>
                <ct:SymbRef symbIdRef="psi_0"/>
            </ct:AssignStatement>
        </ct:BoundaryCondition>

        <ct:BoundaryCondition type="Robin">
            <ct:ConditionVariable>
                <ct:SymbRef symbIdRef="x"/>
            </ct:ConditionVariable>
            <ct:BoundaryValue>
                <ct:Assign>
                    <ct:SymbRef symbIdRef="L"/>
                </ct:Assign>
            </ct:BoundaryValue>
            <ct:AssignStatement op="eq">
                <math:Binop op="plus">
                    <math:PartialDiff>
                        <math:DiffVariable>
                            <ct:SymbRef symbIdRef="x"/>
                        </math:DiffVariable>
                        <math:DiffOpArgument>
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="u"/>
                            </ct:Assign>
                        </math:DiffOpArgument>
                    </math:PartialDiff>
                    <math:Binop op="times">
                        <ct:SymbRef symbIdRef="h"/>
                        <ct:SymbRef symbIdRef="u"/>
                    </math:Binop>
                </math:Binop>
                <ct:SymbRef symbIdRef="psi_L"/>
            </ct:AssignStatement>
        </ct:BoundaryCondition>
    </ct:DE>
```

3.7.6 Reaction-diffusion system – Gray-Scott Model

Source: `http://mrob.com/pub/comp/xmorphia/#formula`

Model definition

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u - uv^2 + F(1-u)$$
$$\frac{\partial v}{\partial t} = D_v \nabla^2 v + uv^2 - (F+k)v$$

Differential equation

```
<StructuralModel blkId="sm1">

   <ct:Variable symbId="u"/>
   <ct:Variable symbId="v"/>
   <!-- partial u/partial t = Du laplacian(u) -uv^2 + F(1-u) -->
   <ct:DE type="pde">
       <ct:AssignStatement op="eq">
           <math:PartialDiff>
               <math:DiffVariable>
                   <ct:SymbRef symbIdRef="t"/>
               </math:DiffVariable>
               <math:DiffOpArgument>
                   <ct:Assign>
                       <ct:SymbRef symbIdRef="u"/>
                   </ct:Assign>
               </math:DiffOpArgument>
           </math:PartialDiff>
           <math:Binop op="plus">
               <math:Binop op="minus">
                   <math:Binop op="times">
                       <ct:SymbRef symbIdRef="Du"/>
                       <math:VectorCalcOp op="laplacian">
                           <math:DiffVariables>
                               <ct:Assign>
                                   <ct:SymbRef symbIdRef="x"/>
                               </ct:Assign>
                           </math:DiffVariables>
                           <math:DiffOpArgument>
                               <ct:Assign>
                                   <ct:SymbRef symbIdRef="u"/>
                               </ct:Assign>
                           </math:DiffOpArgument>
                       </math:VectorCalcOp>
                   </math:Binop>
                   <math:Binop op="times">
                       <ct:SymbRef symbIdRef="u"/>
                       <math:Binop op="power">
                           <ct:SymbRef symbIdRef="v"/>
                           <ct:Real>2</ct:Real>
                       </math:Binop>
                   </math:Binop>
               </math:Binop>
               <math:Binop op="times">
                   <ct:SymbRef symbIdRef="F"/>
                   <math:Binop op="minus">
                       <ct:Real>1</ct:Real>
                       <ct:SymbRef symbIdRef="u"/>
                   </math:Binop>
               </math:Binop>
           </math:Binop>
       </ct:AssignStatement>
   </ct:DE>
   <!-- partial v/partial t = Dv laplacian(v) + uv^2 + (F+k)v) -->
   <ct:DE>
       <ct:AssignStatement op="eq">
           <math:PartialDiff>
               <math:DiffVariable>
                   <ct:SymbRef symbIdRef="t"/>
               </math:DiffVariable>
               <math:DiffOpArgument>
                   <ct:Assign>
                       <ct:SymbRef symbIdRef="v"/>
                   </ct:Assign>
               </math:DiffOpArgument>
           </math:PartialDiff>
           <math:Binop op="minus">
               <math:Binop op="plus">
                   <math:Binop op="times">
                       <ct:SymbRef symbIdRef="Dv"/>
                       <math:VectorCalcOp op="laplacian">
                           <math:DiffVariables>
                               <ct:Assign>
```

```
                                <ct:SymbRef symbIdRef="x"/>
                            </ct:Assign>
                        </math:DiffVariables>
                        <math:DiffOpArgument>
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="v"/>
                            </ct:Assign>
                        </math:DiffOpArgument>
                    </math:VectorCalcOp>
                </math:Binop>
                <math:Binop op="times">
                    <ct:SymbRef symbIdRef="u"/>
                    <math:Binop op="power">
                        <ct:SymbRef symbIdRef="v"/>
                        <ct:Real>2</ct:Real>
                    </math:Binop>
                </math:Binop>
            </math:Binop>
            <math:Binop op="times">
                <math:Binop op="plus">
                    <ct:SymbRef symbIdRef="F"/>
                    <ct:SymbRef symbIdRef="k"/>
                </math:Binop>
                <ct:SymbRef symbIdRef="v"/>
            </math:Binop>
        </math:Binop>
    </ct:AssignStatement>
</ct:DE>
</StructuralModel>
```

3.7.7 Capillary-Tissue Exchange: Convention, Permeation, Reaction, and Diffusion

Source: [Bassingthwaighte et al., 2012]

Model definition

$$\frac{\partial C_p(x,t)}{\partial t} = \frac{F_{pl}L}{V_{pl}}\frac{\partial C_{pl}}{\partial x} + \frac{PS_g}{V_{pl}}\left(C_{pl} - C_{isf}\right) + D_{x1}\frac{\partial^2 C_p}{\partial x^2}$$
$$\frac{\partial C_{isf}(x,t)}{\partial t} = \frac{PS_g}{V_{pl}}\left(C_{pl} - C_{isf}\right) - \frac{G_{isf}}{V_{isf}}C_{isf} + D_{x2}\frac{\partial^2 C_{isf}}{\partial x^2}$$

Differential equation

```
<StructuralModel blkId="sm1">

    <ct:Variable symbId="C_p"/>
    <ct:Variable symbId="C_isf"/>
    <ct:DE type="pde">
        <ct:AssignStatement op="eq">
            <math:PartialDiff>
                <math:DiffVariable>
                    <ct:SymbRef symbIdRef="t"/>
                </math:DiffVariable>
                <math:DiffOpArgument>
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="C_p"/>
                    </ct:Assign>
                </math:DiffOpArgument>
            </math:PartialDiff>
            <math:Binop op="plus">
                <math:Binop op="times">
                    <math:Binop op="divide">
                        <math:Binop op="times">
                            <ct:SymbRef symbIdRef="F_pl"/>
                            <ct:SymbRef symbIdRef="L"/>
                        </math:Binop>
```

```
                    <ct:SymbRef symbIdRef="V_pl"/>
                </math:Binop>
                <math:PartialDiff>
                    <math:DiffVariable>
                        <ct:SymbRef symbIdRef="x"/>
                    </math:DiffVariable>
                    <math:DiffOpArgument>
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="C_pl"/>
                        </ct:Assign>
                    </math:DiffOpArgument>
                </math:PartialDiff>
            </math:Binop>
            <math:Binop op="plus">
                <math:Binop op="times">
                    <math:Binop op="divide">
                        <ct:SymbRef symbIdRef="PS_g"/>
                        <ct:SymbRef symbIdRef="V_pl"/>
                    </math:Binop>
                    <math:Binop op="minus">
                        <ct:SymbRef symbIdRef="C_pl"/>
                        <ct:SymbRef symbIdRef="C_isf"/>
                    </math:Binop>
                </math:Binop>
                <math:Binop op="times">
                    <ct:SymbRef symbIdRef="D_x1"/>
                    <math:PartialDiff>
                        <math:Degree>
                            <ct:Assign>
                                <ct:Int>2</ct:Int>
                            </ct:Assign>
                        </math:Degree>
                        <math:DiffVariable>
                            <ct:SymbRef symbIdRef="x"/>
                        </math:DiffVariable>
                        <math:DiffOpArgument>
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="C_p"/>
                            </ct:Assign>
                        </math:DiffOpArgument>
                    </math:PartialDiff>
                </math:Binop>
            </math:Binop>
        </math:Binop>
    </ct:AssignStatement>

    <!-- 2nd PDE forC_isf is skipped for brevity -->
```

Boundary conditions

$$
\begin{aligned}
BC, x = x_{min}: &\quad \Big(- F_p \frac{L}{V_p} \Big) \big(C_p(x,t) - C_{in}(x,t) \big) + D_p \frac{\partial C_p(x,t)}{\partial x} = 0 \\
BC, x = x_{max}: &\quad \frac{\partial C_p(x,t)}{\partial x} = 0 \\
BC, x = x_{max}: &\quad C_{out}(x,t) = C_p(x,t)
\end{aligned}
$$

```
    <!-- BC (x=x.min): (-F_p*L/V_p)(C_p - C_{in}) + D_p partial C_p/partial x = 0 -->
    <ct:BoundaryCondition>
        <ct:ConditionVariable>
            <ct:SymbRef symbIdRef="x"/>
        </ct:ConditionVariable>
        <ct:BoundaryValue>
            <ct:Assign>
                <ct:SymbRef symbIdRef="xmin"/>
            </ct:Assign>
        </ct:BoundaryValue>
        <ct:AssignStatement op="eq">
            <math:Binop op="plus">
                <math:Binop op="times">
                    <math:Binop op="times">
```

```
                                <math:Uniop op="minus">
                                    <ct:SymbRef symbIdRef="F_p"/>
                                </math:Uniop>
                                <math:Binop op="divide">
                                    <ct:SymbRef symbIdRef="L"/>
                                    <ct:SymbRef symbIdRef="V_p"/>
                                </math:Binop>
                            </math:Binop>
                            <math:Binop op="minus">
                                <ct:SymbRef symbIdRef="C_p"/>
                                <ct:SymbRef symbIdRef="C_in"/>
                            </math:Binop>
                        </math:Binop>
                        <math:Binop op="times">
                            <ct:SymbRef symbIdRef="D_p"/>
                            <math:PartialDiff>
                                <math:DiffVariable>
                                    <ct:SymbRef symbIdRef="x"/>
                                </math:DiffVariable>
                                <math:DiffOpArgument>
                                    <ct:Assign>
                                        <ct:SymbRef symbIdRef="C_p"/>
                                    </ct:Assign>
                                </math:DiffOpArgument>
                            </math:PartialDiff>
                        </math:Binop>
                    </math:Binop>
                    <ct:Real>0</ct:Real>
                </ct:AssignStatement>
            </ct:BoundaryCondition>

            <!-- BC (x=x.max): \frac{\partial C_p}{\partial x} = 0 -->
            <ct:BoundaryCondition>
                <ct:ConditionVariable>
                    <ct:SymbRef symbIdRef="x"/>
                </ct:ConditionVariable>
                <ct:BoundaryValue>
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="xmax"/>
                    </ct:Assign>
                </ct:BoundaryValue>
                <ct:AssignStatement op="eq">
                    <math:PartialDiff>
                        <math:DiffVariable>
                            <ct:SymbRef symbIdRef="x"/>
                        </math:DiffVariable>
                        <math:DiffOpArgument>
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="C_p"/>
                            </ct:Assign>
                        </math:DiffOpArgument>
                    </math:PartialDiff>
                    <ct:Real>0</ct:Real>
                </ct:AssignStatement>
            </ct:BoundaryCondition>

            <!-- BC (x=x.max): C_{out} = C_p -->
            <ct:BoundaryCondition>
                <ct:ConditionVariable>
                    <ct:SymbRef symbIdRef="x"/>
                </ct:ConditionVariable>
                <ct:BoundaryValue>
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="xmax"/>
                    </ct:Assign>
                </ct:BoundaryValue>
                <ct:AssignStatement op="eq">
                    <ct:SymbRef symbIdRef="C_out"/>
                    <ct:SymbRef symbIdRef="C_p"/>
                </ct:AssignStatement>
            </ct:BoundaryCondition>
        </ct:DE>
```

```
        </StructuralModel>
```

3.7.8 Pattern Formation – Schnakenberg system

The following example is form the XPPAUT tutorial (`http://www.math.pitt.edu/~bard/xpp/help/xppexample.html#pde`) and describes apart from the PDE equations also its discretization and encoding as an ODE system.

Model definition

$$\frac{\partial u}{\partial t} = -u + \nu u^2 + d_u \frac{\partial u}{\partial x}$$
$$\frac{\partial \nu}{\partial t} = a - \nu u^2 + d_\nu \frac{\partial \nu}{\partial x}$$

In discretised form

```
u'_j = -u_j + v_j*u^2_j + (d_u/h^2)*(u_j+1 - 2*u_j + u_j-1)
v'j = a - v_j*u^2_j + (d_v/h^2)*(v_j+1 - 2*v_j + v_j-1)
```

and in finally in XPP code as system of ODEs:

```
# schnakenberg PDE 100 points
f(u,v)=-u+v*u^2
g(u,v)=a-v*u^2
u0'=f(u0,v0)+duh*(u1-u0)
u[1..99]'=f(u[j],v[j])+duh*(u[j+1]-2*u[j]+u[j-1])
u100'=f(u100,v100)+duh*(u99-u100)
v0'=g(u0,v0)+dvh*(v1-v0)
v[1..99]'=g(u[j],v[j])+dvh*(v[j+1]-2*v[j]+v[j-1])
v100'=g(u100,v100)+dvh*(v99-v100)
par a=1.05,du=.2,dv=3,h=.2
!duh=du/(h*h)
!dvh=dv/(h*h)
init u0=1.5,u1=1.3,u2=1.1,v0=1.05,v1=1.05,v2=1.05
init u[3..100]=1.05,v[j]=1.05
@ dt=.1,nout=50,total=50,meth=cvode,tol=1e-5,atol=1e-5
done
```

Differential equations in PharmML

```
<StructuralModel blkId="sm1">

    <ct:Variable symbId="u"/>
    <ct:Variable symbId="v"/>

    <ct:DE>
        <ct:AssignStatement op="eq">
            <math:PartialDiff>
                <math:DiffVariable>
                    <ct:SymbRef symbIdRef="t"/>
                </math:DiffVariable>
                <math:DiffOpArgument>
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="u"/>
                    </ct:Assign>
                </math:DiffOpArgument>
            </math:PartialDiff>
            <math:Binop op="plus">
                <math:Uniop op="minus">
                    <ct:SymbRef symbIdRef="u"/>
                </math:Uniop>
```

```
                    <math:Binop op="plus">
                        <math:Binop op="times">
                            <ct:SymbRef symbIdRef="v"/>
                            <math:Binop op="power">
                                <ct:SymbRef symbIdRef="u"/>
                                <ct:Real>2</ct:Real>
                            </math:Binop>
                        </math:Binop>
                        <math:Binop op="times">
                            <ct:SymbRef symbIdRef="d_v"/>
                            <math:PartialDiff>
                                <math:DiffVariable>
                                    <ct:SymbRef symbIdRef="x"/>
                                </math:DiffVariable>
                                <math:DiffOpArgument>
                                    <ct:Assign>
                                        <ct:SymbRef symbIdRef="u"/>
                                    </ct:Assign>
                                </math:DiffOpArgument>
                            </math:PartialDiff>
                        </math:Binop>
                    </math:Binop>
                </math:Binop>
            </ct:AssignStatement>
        </ct:DE>

        <ct:DE>
            <ct:AssignStatement op="eq">
                <math:PartialDiff>
                    <math:DiffVariable>
                        <ct:SymbRef symbIdRef="t"/>
                    </math:DiffVariable>
                    <math:DiffOpArgument>
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="v"/>
                        </ct:Assign>
                    </math:DiffOpArgument>
                </math:PartialDiff>
                <math:Binop op="plus">
                    <math:Binop op="minus">
                        <ct:SymbRef symbIdRef="a"/>
                        <math:Binop op="times">
                            <ct:SymbRef symbIdRef="v"/>
                            <math:Binop op="power">
                                <ct:SymbRef symbIdRef="u"/>
                                <ct:Real>2</ct:Real>
                            </math:Binop>
                        </math:Binop>
                    </math:Binop>
                    <math:Binop op="times">
                        <ct:SymbRef symbIdRef="d_v"/>
                        <math:PartialDiff>
                            <math:DiffVariable>
                                <ct:SymbRef symbIdRef="x"/>
                            </math:DiffVariable>
                            <math:DiffOpArgument>
                                <ct:Assign>
                                    <ct:SymbRef symbIdRef="v"/>
                                </ct:Assign>
                            </math:DiffOpArgument>
                        </math:PartialDiff>
                    </math:Binop>
                </math:Binop>
            </ct:AssignStatement>
        </ct:DE>
    </StructuralModel>
```

Other sources: `http://wwwf.imperial.ac.uk/~hharring/research/finalpatterns.pdf`

3.7.9 Breast cancer development – Enderling et al. 2007

Source: [Enderling et al., 2007]. ODE of the model are omitted. Shown are only the last two, PDEs for tumour cells and the enzymes which they produce, denoted by n and m variables, respectively.

Model definition

$$\frac{df}{dt} = -\overbrace{\kappa_f m f}^{\text{degradation}} - \overbrace{\vartheta f}^{f \to \mathrm{TSG}_1^{+/-}},$$

$$\frac{dq}{dt} = \overbrace{\mu_q q(A_{\max} - A)}^{\text{proliferation}} - \overbrace{\kappa_q n q}^{\text{death}} + \overbrace{\vartheta f}^{f \to \mathrm{TSG}_1^{+/-}} - \overbrace{p_2 q}^{\mathrm{TSG}_1^{+/-} \to \mathrm{TSG}_1^{-/-}},$$

$$\frac{dr}{dt} = \overbrace{\mu_r r(A_{\max} - A)}^{\text{proliferation}} - \overbrace{\kappa_r n r}^{\text{death}} + \overbrace{p_2 q}^{\mathrm{TSG}_1^{+/-} \to \mathrm{TSG}_1^{-/-}} - \overbrace{p_3 r}^{\mathrm{TSG}_1^{-/-} \to \mathrm{TSG}_2^{+/-}},$$

$$\frac{ds}{dt} = \overbrace{\mu_s s(A_{\max} - A)}^{\text{proliferation}} - \overbrace{\kappa_s n s}^{\text{death}} + \overbrace{p_3 r}^{\mathrm{TSG}_1^{-/-} \to \mathrm{TSG}_2^{+/-}} - \overbrace{p_4 s}^{\mathrm{TSG}_2^{+/-} \to n},$$

$$\frac{\partial n}{\partial t} = \overbrace{\mu_n n(A_{\max} - A)}^{\text{proliferation}} + \overbrace{D_n \nabla^2 n}^{\text{random motility}} - \overbrace{\chi \nabla \cdot (n \nabla f)}^{\text{haptotaxis}} + \overbrace{p_4 s}^{\mathrm{TSG}_2^{+/-} \to n},$$

$$\frac{\partial m}{\partial t} = \overbrace{D_m \nabla^2 m}^{\text{diffusion}} + \overbrace{\zeta n(1 - m/m_0)}^{\text{production}} - \overbrace{\omega m}^{\text{decay}},$$

Figure 3.1: Screenshot from the paper [Enderling et al., 2007]. Implemented are only the last two PDEs.

```
<!-- PDE: delta n / delta t = ... -->
<ct:DE type="pde">
    <ct:AssignStatement op="eq">
        <math:PartialDiff>
            <math:DiffVariable>
                <ct:SymbRef symbIdRef="t"/>
            </math:DiffVariable>
            <math:DiffOpArgument>
                <ct:Assign>
                    <ct:SymbRef symbIdRef="n"/>
                </ct:Assign>
            </math:DiffOpArgument>
        </math:PartialDiff>
        <!-- RHS -->
        <math:Binop op="plus">
            <math:Binop op="times">
                <ct:SymbRef blkIdRef="pm1" symbIdRef="mu_n"/>
                <math:Binop op="times">
                    <ct:SymbRef symbIdRef="n"/>
                    <math:Binop op="minus">
                        <ct:SymbRef blkIdRef="pm1" symbIdRef="Amax"/>
                        <ct:SymbRef blkIdRef="pm1" symbIdRef="A"/>
                    </math:Binop>
                </math:Binop>
            </math:Binop>
            <math:Binop op="plus">
                <math:Binop op="times">
                    <ct:SymbRef symbIdRef="D_n"/>
                    <math:VectorCalcOp op="laplacian">
                        <math:DiffVariables>
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="x"/>
                            </ct:Assign>
                        </math:DiffVariables>
                        <math:DiffOpArgument>
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="n"/>
                            </ct:Assign>
```

```
                        </math:DiffOpArgument>
                    </math:VectorCalcOp>
                </math:Binop>
                <math:Binop op="plus">
                    <math:Uniop op="minus">
                        <math:Binop op="times">
                            <ct:SymbRef blkIdRef="pm1" symbIdRef="chi"/>
                            <math:VectorCalcOp op="divergence">
                                <math:DiffOpArgument>
                                    <ct:Assign>
                                        <math:Binop op="times">
                                            <ct:SymbRef symbIdRef="n"/>
                                            <math:VectorCalcOp op="gradient">
                                                <math:DiffVariables>
                                                    <ct:Assign>
                                                        <ct:SymbRef symbIdRef="x"/>
                                                    </ct:Assign>
                                                </math:DiffVariables>
                                                <math:DiffOpArgument>
                                                    <ct:Assign>
                                                        <ct:SymbRef symbIdRef="f"/>
                                                    </ct:Assign>
                                                </math:DiffOpArgument>
                                            </math:VectorCalcOp>
                                        </math:Binop>
                                    </ct:Assign>
                                </math:DiffOpArgument>
                            </math:VectorCalcOp>
                        </math:Binop>
                    </math:Uniop>
                    <math:Binop op="times">
                        <ct:SymbRef blkIdRef="pm1" symbIdRef="p4"/>
                        <ct:SymbRef symbIdRef="s"/>
                    </math:Binop>
                </math:Binop>
            </math:Binop>
        </math:Binop>
    </ct:AssignStatement>
</ct:DE>

<!-- PDE: delta m / delta t = ... -->
<ct:DE type="pde">
    <ct:AssignStatement op="eq">
        <math:Diff>
            <math:DiffVariable>
                <ct:SymbRef symbIdRef="t"/>
            </math:DiffVariable>
            <math:DiffOpArgument>
                <ct:Assign>
                    <ct:SymbRef symbIdRef="m"/>
                </ct:Assign>
            </math:DiffOpArgument>
        </math:Diff>
        <math:Binop op="plus">
            <math:Binop op="times">
                <ct:SymbRef symbIdRef="D_m"/>
                <math:VectorCalcOp op="laplacian">
                    <math:DiffVariables>
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="x"/>
                        </ct:Assign>
                    </math:DiffVariables>
                    <math:DiffOpArgument>
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="m"/>
                        </ct:Assign>
                    </math:DiffOpArgument>
                </math:VectorCalcOp>
            </math:Binop>
            <math:Binop op="minus">
                <math:Binop op="times">
                    <ct:SymbRef symbIdRef="xi"/>
```

```
                        <math:Binop op="times">
                            <ct:SymbRef symbIdRef="n"/>
                            <math:Binop op="minus">
                                <ct:Real>1</ct:Real>
                                <math:Binop op="divide">
                                    <ct:SymbRef symbIdRef="m"/>
                                    <ct:SymbRef symbIdRef="m_0"/>
                                </math:Binop>
                            </math:Binop>
                        </math:Binop>
                        <math:Binop op="times">
                            <ct:SymbRef blkIdRef="pm1" symbIdRef="omega"/>
                            <ct:SymbRef symbIdRef="m"/>
                        </math:Binop>
                    </math:Binop>
                </math:Binop>
            </ct:AssignStatement>
        </ct:DE>
```

3.7.10 PharmML code

All models listed in this section are fully encoded and available in the release pack: *Advection.xml*, *Advection-Diffusion.xml*, *capillaryTissueExchange.xml*, *Diffusion_Ghaffarizadeh.xml*, *Diffusion.xml*, *Enderling2007.xml*, *SchnakenbergSystem.xml*, *ReactionDiffusion.xml*, *Heat.xml*.

Chapter 4

New use cases

4.1 Handling IOV in optimal design models

New challenges occurred when we started to encode the optimal design test cases with inter-occasion (aka between-occasion) variability. It turns out there are several ways to encode it, depending on what type of mapping is used and where it is implemented. For this discussion we consider a standard setup illustrated in Figure 4.1 with two arms, two occasions and design related covariates: treatments and sequence of treatments.

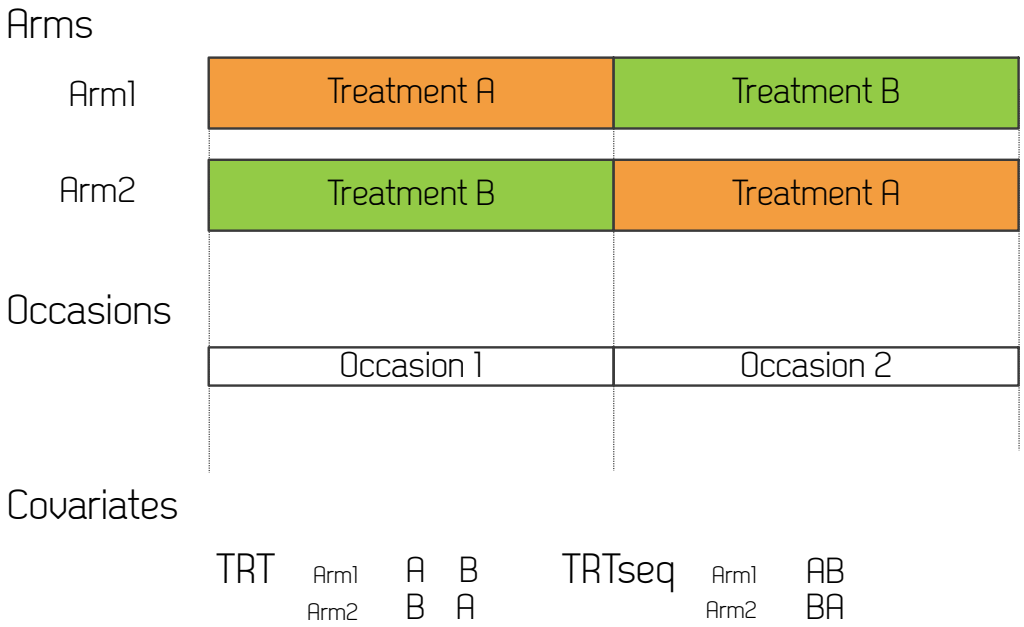


Figure 4.1: IOV with design related covariates.

4.1.1 Implementation options

Option 1 Association of covariates-occasions within arms

Option 2 Arm-specific occasions declaration and association with covariates

Option 3 Covariates and occasions declared in the top level

Option 4 Covariates and occasions declared in `<IndividualCovariates>`

4.1.2 Option 1

Define

- occasions for all arms in the top level of `<TrialDesign>` with IOV reference

```
<Occasions>
    <OccasionList oid="ol1">
        <ct:VariabilityReference>
            <ct:SymbRef blkIdRef="vm_mdl" symbIdRef="OCC"/>
        </ct:VariabilityReference>
        <Occasion oid="OCC_1">
            <Start>
                <ct:Assign>
                    <ct:Int>0</ct:Int>
                </ct:Assign>
            </Start>
        </Occasion>
        <Occasion oid="OCC_2">
            <Start>
                <ct:Assign>
                    <ct:Int>120</ct:Int>
                </ct:Assign>
            </Start>
        </Occasion>
    </OccasionList>
</Occasions>
```

- the occasion sequence is then referenced from each arm

```
        <Arm oid="arm1">
            <CovariateModel>
                <!-- omitted details -->
            </CovariateModel>
            <InterventionSequence>
                <!-- omitted details -->
            </InterventionSequence>
            <ObservationSequence>
                <!-- omitted details -->
            </ObservationSequence>

            <OccasionSequence>
                <OccasionListRef oidRef="ol1"/>
            </OccasionSequence>
        </Arm>
```

- declare also the `TRT` covariate in each arm and associate its categories with occasions

```
    <Arms>
        <Arm oid="arm1">
            <CovariateModel oid="arm1_td_cm">
                <CovariateModelRef oidRef="cm"/>
                <Covariate symbId="TRT">
                    <mdef:Categorical>
                        <mdef:Category catId="A">
                            <mdef:OccasionRef oidRef="OCC_1"/>
                        </mdef:Category>
                        <mdef:Category catId="B">
                            <mdef:OccasionRef oidRef="OCC_2"/>
                        </mdef:Category>
                    </mdef:Categorical>
                </Covariate>
                <Covariate symbId="TRTseq">
                    <mdef:Categorical>
                        <mdef:Category catId="AB"/>
                    </mdef:Categorical>
                </Covariate>
            </CovariateModel>
```

4.1.3 Option 2

- Declare arm-specific occasions (even though they are defined on the same time intervals) with different object identifiers, `oid`, e.g. *arm1_OCC_1* and *arm1_OCC_2* for arm 1

```
<Arms>
    <Arm oid="arm1">
        <OccasionSequence>
            <OccasionList oid="arm1_OCC">
                <ct:VariabilityReference>
                    <ct:SymbRef blkIdRef="vm_mdl" symbIdRef="OCC"/>
                </ct:VariabilityReference>
                <Occasion oid="arm1_OCC_1">
                    <Start>
                        <ct:Assign>
                            <ct:Int>0</ct:Int>
                        </ct:Assign>
                    </Start>
                </Occasion>
                <Occasion oid="arm1_OCC_2">
                    <Start>
                        <ct:Assign>
                            <ct:Int>120</ct:Int>
                        </ct:Assign>
                    </Start>
                </Occasion>
            </OccasionList>
        </OccasionSequence>
    </Arm>
```

- associate these occasions to categories as well in each arm

```
    <CovariateModel oid="arm2_td_cm">
        <CovariateModelRef oidRef="cm"/>
        <Covariate symbId="TRT">
            <mdef:Categorical>
                <mdef:Category catId="B">
                    <mdef:OccasionRef oidRef="arm2_OCC_1"/>
                </mdef:Category>
                <mdef:Category catId="A">
                    <mdef:OccasionRef oidRef="arm2_OCC_2"/>
                </mdef:Category>
            </mdef:Categorical>
        </Covariate>
        <Covariate symbId="TRTseq">
            <mdef:Categorical>
                <mdef:Category catId="BA"/>
            </mdef:Categorical>
        </Covariate>
    </CovariateModel>
```

4.1.4 Option 3

In this case occasions are defined in the top trial design level and their sequences referenced in each arm. Also the covariate model is located in the top level with

- the TRT covariate is associated with administrations, here `admin1` and `admin2`

```
<Covariates>
    <CovariateModel oid="td_cm">
        <CovariateModelRef oidRef="cm"/>
        <Covariate symbId="TRT">
            <mdef:Categorical>
                <mdef:Category catId="B">
                    <mdef:InterventionRef oidRef="admin1"/>
                </mdef:Category>
                <mdef:Category catId="A">
                    <mdef:InterventionRef oidRef="admin2"/>
                </mdef:Category>
            </mdef:Categorical>
        </Covariate>
```

- while the TRTseq covariate declared conditional on arms

```
                    <Covariate symbId="TRTseq">
                        <mdef:Categorical>
                            <ct:Assign>
                                <math:Piecewise>
                                    <math:Piece>
                                        <ct:CatRef catIdRef="AB"/>
                                        <math:Condition>
                                            <math:LogicBinop op="eq">
                                                <StudyArm/>
                                                <ct:OidRef oidRef="arm1"/>
                                            </math:LogicBinop>
                                        </math:Condition>
                                    </math:Piece>
                                    <math:Piece>
                                        <ct:CatRef catIdRef="BA"/>
                                        <math:Condition>
                                            <math:LogicBinop op="eq">
                                                <StudyArm/>
                                                <ct:OidRef oidRef="arm2"/>
                                            </math:LogicBinop>
                                        </math:Condition>
                                    </math:Piece>
                                </math:Piecewise>
                            </ct:Assign>
                        </mdef:Categorical>
                    </Covariate>
                </CovariateModel>
            </Covariates>
```

4.1.5 Option 4

This case is different from the previous ones in that the <IndividualCovariates> is used and the relevant design characteristic are stored within inline dataset. The following code snippet shows the columns definition and complete data record

```
<Covariates>
    <IndividualCovariates oid="ic1">
        <ds:DataSet>
            <ColumnMapping>
            <!-- see below -->
            </ColumnMapping>
            <ds:Definition>
                <ds:Column columnId="ARM" columnType="arm" valueType="id" columnNum="1"/>
                <ds:Column columnId="TIME" columnType="time" valueType="real" columnNum="2"/>
                <ds:Column columnId="OCC" columnType="varLevel" valueType="string" columnNum="3"/>
                <ds:Column columnId="TRT" columnType="covariate" valueType="string" columnNum="4"/>
                <ds:Column columnId="TRTseq" columnType="covariate" valueType="string" columnNum="5"/>
            </ds:Definition>
            <ds:Table>
                <ds:Row>
                    <ct:Id>arm1</ct:Id><ct:Real>0</ct:Real><ct:String>occ1</ct:String><ct:String>A</ct:String><ct:String>AB</ct:S
                    <ct:Id>arm1</ct:Id><ct:Real>119</ct:Real><ct:String>occ1</ct:String><ct:String>A</ct:String><ct:String>AB</ct
                    <ct:Id>arm1</ct:Id><ct:Real>120</ct:Real><ct:String>occ2</ct:String><ct:String>B</ct:String><ct:String>AB</ct
                    <ct:Id>arm1</ct:Id><ct:Real>240</ct:Real><ct:String>occ2</ct:String><ct:String>B</ct:String><ct:String>AB</ct
                </ds:Row>
                <ds:Row>
                    <ct:Id>arm2</ct:Id><ct:Real>0</ct:Real><ct:String>occ1</ct:String><ct:String>B</ct:String><ct:String>BA</ct:S
                    <ct:Id>arm2</ct:Id><ct:Real>119</ct:Real><ct:String>occ1</ct:String><ct:String>B</ct:String><ct:String>BA</ct
                    <ct:Id>arm2</ct:Id><ct:Real>120</ct:Real><ct:String>occ2</ct:String><ct:String>A</ct:String><ct:String>BA</ct
                    <ct:Id>arm2</ct:Id><ct:Real>240</ct:Real><ct:String>occ2</ct:String><ct:String>A</ct:String><ct:String>BA</ct
                </ds:Row>
            </ds:Table>
        </ds:DataSet>
```

Note that the columnType attribute for occasion column OCC has the value *varLevel* indicating that it is variability level related one and the mapping makes sure that it is related to the iov1 level defined the variability model. If occasions are to be considered as covariates as well the columnType can handle multiple assignments and would read in such case columnType="varLevel covariate". (In this case a mapping to the covariate model would be required as well.)

The rest is achieved by appropriate <ColumnMapping>, i.e.

```
<ColumnMapping> <!-- IOV1 mapping -->
    <ds:ColumnRef columnIdRef="OCC"/>
    <ct:SymbRef blkIdRef="vm_mdl" symbIdRef="OCC"/>
</ColumnMapping>
<ColumnMapping>
    <ds:ColumnRef columnIdRef="TRT"/>
    <ct:SymbRef blkIdRef="cm" symbIdRef="TRT"/>
    <ds:CategoryMapping>
        <ds:Map dataSymbol="A" modelSymbol="A"/>
        <ds:Map dataSymbol="B" modelSymbol="B"/>
    </ds:CategoryMapping>
</ColumnMapping>
<ColumnMapping>
    <ds:ColumnRef columnIdRef="TRTseq"/>
    <ct:SymbRef blkIdRef="cm" symbIdRef="TRTseq"/>
    <ds:CategoryMapping>
        <ds:Map dataSymbol="AB" modelSymbol="AB"/>
        <ds:Map dataSymbol="BA" modelSymbol="BA"/>
    </ds:CategoryMapping>
</ColumnMapping>
```

4.2 Constraints

This example comes from [Dalla Man et al., 2002] and illustrates how constraints can be declared.

$$\int_0^{420} Ra\ dx = \frac{Df}{BW}$$

```
<ct:AssignStatement op="eq">
    <math:Integral>
        <math:LowLimit>
            <ct:Assign>
                <ct:Real>0</ct:Real>
            </ct:Assign>
        </math:LowLimit>
        <math:UpLimit>
            <ct:Assign>
                <ct:Real>420</ct:Real>
            </ct:Assign>
        </math:UpLimit>
        <math:IntegrationVariable>
            <ct:Assign>
                <ct:SymbRef symbIdRef="t"/>
            </ct:Assign>
        </math:IntegrationVariable>
        <math:IntegralArgument>
            <ct:Assign>
                <ct:SymbRef symbIdRef="Ra"/>
            </ct:Assign>
        </math:IntegralArgument>
    </math:Integral>
    <math:Binop op="divide">
        <math:Binop op="times">
            <ct:SymbRef symbIdRef="D"/>
            <ct:SymbRef symbIdRef="f"/>
        </math:Binop>
        <ct:SymbRef blkIdRef="cm1" symbIdRef="BW"/>
    </math:Binop>
</ct:AssignStatement>
```

Bibliography

[Bassingthwaighte et al., 2012] Bassingthwaighte, J. B., Butterworth, E., Jardine, B., and Raymond, G. M. (2012). *Computational Toxicology: Volume I*, chapter Compartmental Modeling in the Analysis of Biological Systems, pages 391–438. Humana Press, Totowa, NJ.

[Beal, 2001] Beal, S. L. (2001). Ways to fit a pk model with some data below the quantification limit. *Journal of Pharmacokinetics and Pharmacodynamics*, 28(5):481–504.

[Bonate, 2011] Bonate, P. L. (2011). *Pharmacokinetic-pharmacodynamic modeling and simulation*. Springer, New York.

[Box and Cox, 1964] Box, G. and Cox, D. (1964). An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 26(2):211–252.

[Comets et al., 2015] Comets, E., Chenel, M., and Hooker, A. (2015). Modelling Description Language, Design elements, Draft 1 version 2. Technical report, INSERM, UPD; Servier; Uppsala University.

[Dalla Man et al., 2002] Dalla Man, C., Caumo, A., and Cobelli, C. (2002). The oral glucose minimal model: Estimation of insulin sensitivity from a meal test. *Ieee Transactions On Biomedical Engineering*, 49(5):419–429.

[Davidian, 2009] Davidian, M. (2009). ST 762, Nonlinear Models for Univariate and Multivariate Response, Lecture Notes.

[Enderling et al., 2007] Enderling, H., Chaplain, M. A. J., Anderson, A. R. A., and Vaidya, J. S. (2007). A mathematical model of breast cancer development, local treatment and recurrence. *J Theor Biol*, 246(2):245–59.

[Ermentrout, 2002] Ermentrout, B. (2002). *Simulating, analyzing, and animating dynamical systems: a guide to XPPAUT for researchers and students*, volume 14. Siam.

[Ghaffarizadeh et al., 2016] Ghaffarizadeh, A., Friedman, S. H., and Macklin, P. (2016). Biofvm: an efficient, parallelized diffusive transport solver for 3-d biological simulations. *Bioinformatics*, 32(8):1256–8.

[Grodsky, 1972] Grodsky, G. M. (1972). A threshold distribution hypothesis for packet storage of insulin and its mathematical modeling. *J Clin Invest*, 51(8):2047–59.

[Lavielle, 2014] Lavielle, M. (2014). *Mixed Effects Models for the Population Approach: Models, Tasks, Methods and Tools*. Chapman & Hall/CRC Biostatistics Series.

[Mager, 2006] Mager, D. E. (2006). Quantitative structure-pharmacokinetic/pharmacodynamic relationships. *Adv Drug Deliv Rev*, 58(12-13):1326–56.

[Raymond et al., 2003] Raymond, G., Butterworth, E., and Bassingthwaighte, J. (2003). Jsim: free software package for teaching physiological modeling and research. In *Faseb Journal*, volume 17, pages A390–A390.

[Sakia, 1992] Sakia, R. M. (1992). The box-cox transformation technique: a review. *The Statistician*, 41:169–178.

[Swat et al., 2016a] Swat, M. J., Grenon, P., and Wimalaratne, S. (2016a). ProbOnto: ontology and knowledge base of probability distributions. *Bioinformatics*.

[Swat et al., 2016b] Swat, M. J., Grenon, P., and Wimalaratne, S. M. (2016b). ProbOnto 1.2 - Ontology and Knowledge Base of Probability Distributions. Technical report, EMBL-EBI, Hinxton, UK; UCL, UK.

[Swat et al., 2015a] Swat, M. J., Grenon, P., Yvon, F., Wimalaratne, S., and Kristensen, N. R. (2015a). Extentions in PharmML 0.7-0.7.2. Technical report, EMBL-EBI.

[Swat et al., 2015b] Swat, M. J., Wimalaratne, S. M., Kristensen, N. R., Yvon, F., Moodie, S., and Le Novère, N. (2015b). Pharmacometrics Markup Language (PharmML), Language Specification for Version 0.6.

[Yngman, 2016] Yngman, G. (2016). Issues in the nt2mdl converter. Technical report, Uppsala University.