Citation: CPT Pharmacometrics Syst. Pharmacol. (2017) 6, 651–665; doi:10.1002/psp4.12213

TUTORIAL

PharmML in Action: an Interoperable Language for Modeling and Simulation

R Bizzotto[1], E Comets[2,3]*, G Smith[4], F Yvon[5], NR Kristensen[6] and MJ Swat[5,7]

PharmML[1] is an XML-based exchange format[2–4] created with a focus on nonlinear mixed-effect (NLME) models used in pharmacometrics,[5,6] but providing a very general framework that also allows describing mathematical and statistical models such as single-subject or nonlinear and multivariate regression models. This tutorial provides an overview of the structure of this language, brief suggestions on how to work with it, and use cases demonstrating its power and flexibility.
CPT Pharmacometrics Syst. Pharmacol. (2017) **6**, 651–665; doi:10.1002/psp4.12213; published online 15 September 2017.

THE PharmML LANGUAGE

Over the past decades, Modeling and Simulation have become a prominent approach to analyze data from preclinical and clinical trials, as well as to better understand the properties of complex systems in biology. The types of data collected in these trials and experiments are extremely variable, in terms of both measured outcomes and designs. This gave rise to new disciplines, in particular pharmacometrics, which is defined as the science of quantitative pharmacology[7] and specializes in mathematical and statistical methods to characterize, quantify, and predict the pharmacokinetics (PK) and the effect of drugs (pharmacodynamics, PD), and quantitative systems pharmacology (QSP), which combines computational and experimental methods to elucidate, validate, and apply new pharmacological concepts to the development and use of small molecules and biologic drugs.[8] A common feature in these different disciplines is the need to define mathematical models, although the meaning and scope of what constitutes a model may vary widely across applications and purpose of the analysis. From a methodological point of view, estimation approaches, simulation methods, and integration algorithms were developed and refined to handle this expanding amount of data and the types of models developed to describe them. They have been implemented in a number of different tools, such as NONMEM[9] and Monolix[10] in pharmacometrics, or Simcyp Simulator in QSP.[11] Because of the way they have evolved, these tools do not necessarily talk to each other, even within a single field, and even less across disciplines. This makes it difficult to perform integrated analyses or even to compare different estimation or simulation approaches on the same application, and limits cross-fertilization across fields. Because the model is at the core of every analysis, the first step to remedy this situation is to define a standard to encode models, which acts as a common hub for each software to connect to, and helps to frame the models in a way that is consistent across disciplines. A similar effort, which started more than 15 years ago, was undertaken in the Systems Biology community and led to the creation of the SBML standard, again an XML-based format developed to encode computational models of biological processes.[12] The SBML standard has fostered a major burst in research and development of new tools[13] and model repositories[14–16] and is an essential component in sustainable model building based on standard formats and open storage of models.[17]

These considerations prompted the creation of the Drug Disease Model Resources (DDMoRe) consortium in 2011, aiming to improve the quality, efficiency, and cost-effectiveness of Model-Informed Drug Discovery & Development[18] (http://ddmore.eu). A major undertaking of the DDMoRe project was the development of a language allowing expressing a large variety of models from both pharmacometrics and QSP, and allowing their encoding in a machine understandable format that could be shared across software through the development of translators and connectors. This language, PharmML, is a declarative language, which is a necessary condition to achieve interoperability among the tools used in pharmacometrics.

The use of XML to define schemas for exchange formats is widely adopted across various industries and scientific areas (https://en.wikipedia.org/wiki/Category:XML-based_standards). It allows defining domain-specific elements in a way which is both human and machine readable. The creation of the PharmML schema required in-depth analysis of pharmacometrics and the models it uses, especially the nonlinear mixed effect models, and was based on textbooks that describe the mathematical and statistical foundations of the domain.[5,6]

PharmML can already interact with major software in the Modeling and Simulation arena, such as the population PK/PD-specific software Monolix[10] and NONMEM,[9] the general statistical software R,[19] and the Bayesian inference software WinBUGS[20] (**Figure 1**). Indeed, within the DDMoRe framework, a number of converters have already been created to automate calls to these "target" software, demonstrating that PharmML can be used as a cornerstone in interoperability. This tutorial is expected to facilitate further development of these and new converters, to foster the

[1]CNR Institute of Neuroscience, Padua, Italy; [2]INSERM, IAME, UMR 1137, Université Paris Diderot, Sorbonne Paris Cité, Paris, France; [3]INSERM CIC 1414, Université de Rennes 1, Rennes, France; [4]Scientific Computing Group, Cyprotex Discovery Ltd, Cheshire, United Kingdom; [5]EMBL-EBI, Wellcome Genome Campus, Hinxton, United Kingdom; [6]Novo Nordisk, Denmark; [7]Simcyp Limited (a Certara company), Sheffield, United Kingdom. *Correspondence: E Comets (emmanuelle.comets@inserm.fr)
Received 28 November 2016; accepted 18 May 2017; published online on 15 September 2017. doi:10.1002/psp4.12213 This article was published online on 15 September 2017. An error was subsequently identified. This notice is included in the online version to indicate the article has been corrected 12 October 2017.

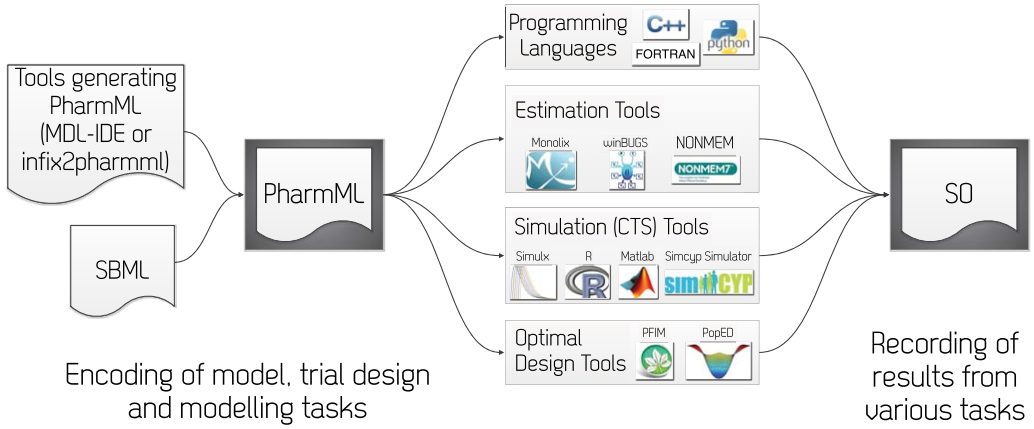

Figure 1 PharmML at the center of interoperability. Once a PharmML model is formulated, using the MDL-IDE or the infix2pharmml tool or translating from SBML, the model can be moved to any PharmML compatible target software and the target output can be translated into a standardized file using the SO format.

development of PharmML editors and graphical tools, and to encourage life scientists to approach this powerful and flexible language in order to make their daily job less prone to language-related bottlenecks and more suitable for cross-fertilization. The development of PharmML went through many different steps (see http://www.pharmml.org/ for an overview), but this tutorial focuses on the scope and features of version 0.8.1.

Scope of PharmML

PharmML supports both continuous and discrete data models and has been validated against a wide range of real-life pharmacometric[21] and life sciences models. A strength of PharmML is that it can be applied to any kind of deterministic model that can be formulated using algebraic, ordinary, and delayed differential equations, covering most models encountered in PK, PD, PK/PD, or QSP. Discrete data models pose a challenge, as they come in a large variety: PharmML provides a rich vocabulary to cover count, categorical, and time-to-event data models, Markovian dependencies, and censoring.[6,22–24] Combinations of any number of observations in one model is permitted. Modeling individual and population parameters is one of the key ingredients in pharmacometrics. The former especially requires a rich and flexible supporting structure. PharmML allows to express fixed and random effects, population values, and covariates in a well-structured form, e.g., as Gaussian models, or to combine them in an arbitrary expression. Continuous and categorical covariates, and intrinsic or extrinsic variables that may influence the PK or PD of a drug,[5,25] are supported, together with the declaration of their distribution, transformation or interpolation.

Overall structure of PharmML

Figure 2 gives a comprehensive view of a full PharmML code, with its different components delineated by colored dotted lines. The language is structured through markup constructs that provide a hierarchical definition of components and subcomponents. Each of them is called block or element (the former allows referencing) and is identified by a pair of beginning and ending tags (e.g., `<RandomEffects>` and `</RandomEffects>`, respectively, within the Parameter Model) or by a unique self-closing tag, ending with / (e.g., `<ct:SymbRef symbIdRef="eta_Psi"/>`, again within the Parameter Model). The markup format provides a structured yet flexible way for content organization.

The description of the structural (i.e., deterministic) model, as well as the parameter, variability, covariate, and observation models, is included in a comprehensive PharmML section called "Model Definition" (**Figure 2**, green box). A second language element is the "Trial Design" section (**Figure 2**, light blue box), which describes inputs to and outputs from the system. Within pharmacometrics this typically encompasses drug treatment regimens and interventions for a subject or groups of subjects, and sampling schedules defining when observations are collected and for which model components; within QSP, the Trial Design can define system perturbations and resulting outputs. On any design element, the set of its possible values, defining a design space, can also be declared for simulation or optimization purposes. A Trial Design can be described explicitly or implicitly, i.e., without or with the aid of external files, respectively. The third PharmML component is the "Modeling Steps" section containing the description of estimation, simulation, or design evaluation/optimization tasks with their characteristic features and the target software settings required for their execution (**Figure 2**, red box). These three components and their main internal blocks are shown schematically in the "**Supplementary information – Figures**" file: the overview is provided by **Figure S1**, while the detailed structure of each block is shown in **Figures S2** to **S4**.

Model Definition, Trial Design and Modeling Steps sections form the backbone of PharmML and underpin its hierarchical structure. The next three chapters of this article address each of them in turn. The chapter **WORKING WITH PharmML** briefly describes useful ways for creating, validating, and converting PharmML codes.

PharmML use cases

PharmML can be used to define a variety of models, trial designs, and tasks. We provide several detailed examples in a supplementary file ("**Supplementary information –**

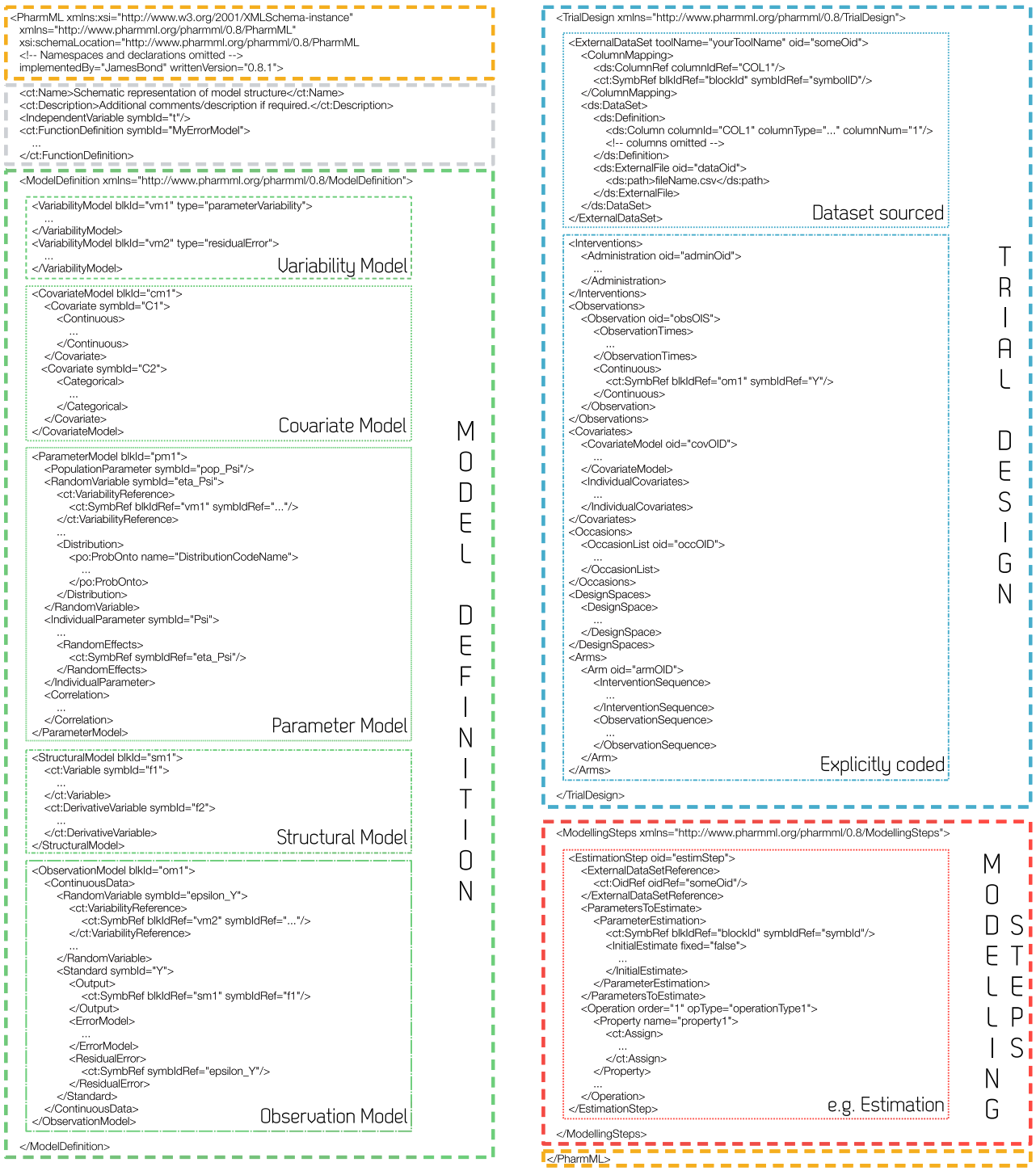

Figure 2 PharmML code example: visualization of the essential elements of the code implementing a basic continuous data model (this is apparent from the observation model, adopting the `<ContinuousData>` tag). Note that for the purpose of illustration, the trial design section (blue) contains both the dataset sourcing (NONMEM/Monolix format) and the explicit design, but usually only one of them is used. The modeling step section (red) declares as an example an estimation task but other tasks are feasible as well. The yellow and gray sections show lines of code providing the header and closure of the code (yellow) and the statement of the name and description of the model, its independent variable and any user-defined functions (gray).

ObservationModel y_{ij}

ContinuousData

Distribution Type $u(y_{ij}) \sim N\big(u(f(x_{ij},\psi_i)),\sigma\big)$

Structured Type $u(y_{ij}) = u(f(x_{ij},\psi_i)) + g(x_{ij},\psi_i,\xi_i)\,\epsilon_{ij}$

Equation Type $u(y_{ij}) = U\big(f(x_{ij},\psi_i)),\xi_i,\epsilon_{1,ij},\epsilon_{2,ij},...\big)$

$$\log(y_{ij}) \sim N\big(\log(f(x_{ij},\psi_i)),a^2\big)$$
$$\log(y_{ij}) = \log(f(x_{ij},\psi_i)) + a^2\,\epsilon_{ij},\quad \epsilon_{ij}\sim N(0,1)$$
$$y_{ij} = f(x_{ij},\psi_i)\exp(a\,\epsilon_{ij}),\quad \epsilon_{ij}\sim N(0,1)$$

Usually
$f(x_{ij},\psi_i) \equiv Cc(t,\psi)$
i.e. drug concentration as predicted by the structural model

CountData

$$y_{ij} \sim Poisson\big(\lambda(t_{ij},\psi_i)\big)$$
$$\lambda(t_{ij},\psi_i) = \lambda_i$$
$$\lambda(t_{ij},\psi_i) = \lambda_{0,i}\Big(1 - Imax_i\frac{f_i(x_{ij},\psi_i)}{IC50_i + f_i(x_{ij},\psi_i)}\Big)$$

CategoricalData

$$y_{ij} \sim Cat\big(\pi_1,\pi_2,...,\pi_K\big)$$
$$\text{logit}\big(P(y_{ij}\le cat1)\big) = \theta_1 + \theta_2\,f(x_{ij},\psi_i)$$

TimeToEventData

$$P(y_i > t;\psi_i) = S(t,\psi_i) = \exp(-H(\tau_0,t;\psi_i))$$
$$H(a,b;\psi_i) = \int_a^b h(t,\psi_i)dt,\quad \text{e.g. } h(t,\beta) = \beta\times f(x_{ij},\psi_i)$$

StructuralModel f

ODE
$$dAd/dt = -k_a\,Ad$$
$$dAc/dt = k_a\,Ad - k\,Ac,\quad C = Ac/V$$

Algebraic Eq. $C = \dfrac{D}{V}\dfrac{k}{k_a-k}\Big(e^{-k\,t} - e^{-k_a\,t}\Big)$

PK macros
$\text{compartment}(\text{amount}=Ac,\text{volume}=V)$
$\text{oral}(ka)$
$\text{elimination}(k)$

ParameterModel ψ_i

Structured Type $h(\psi_i) = h(\psi_{pop}) + \beta\,c_i + \eta$

Distribution Type $h(\psi_i) \sim N\big(h(\psi_{pop}) + \beta\,c_i,\omega^2\big)$

Equation Type $h(\psi_i) = H(\beta,c,\psi_{pop},\eta)$

$$\log(V_i) = \log(V_{pop}) + \beta\,logWT70 + \eta_V$$
$$\log(V_i) \sim N\big(\log(V_{pop}) + \beta\,logWT70,\omega_V^2\big)$$
$$V_i = V_{pop}\big(WT/70\big)^{\beta}\exp(\eta_V)$$

CovariateModel c_i

Distribution $WT \sim LN(\mu_{WT},\sigma^2_{WT})$

Transformation $logWT70 = \log(WT/70)$

Interpolation $linear : WT \text{ versus } Age/Time$

New covariate $CL = \dfrac{(180-Age)\times W\times 0.85}{72\times Scr}$

VariabilityModel η,ϵ

Hierarchical structures for

- Parameter related variability
- Residual error related variability

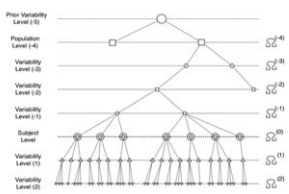

Figure 3 Essential building blocks of a general NLME model. Top: the observation model for continuous data can be defined via three equivalent options, the "structured," "distribution," and "equation" types. Their application to a Gaussian observation model is shown, but other models are allowed as long as ε's are symmetrically distributed with null mean. The shown count data model is a Poisson one, with constant or time varying parameter; other models include negative binomial, generalized Poisson, etc. Nominal and ordered categorical data models can be formulated with a probability mass function and cumulative/tail probabilities per category, respectively. Time-to-event data models can be specified by survival or hazard functions. Middle, left: the structural model prediction can be formulated using ODEs, algebraic equations, or PK macros. Middle, right: there are three equivalent options to encode a parameter model, the "structured," "distribution," and "equation" types: an exemplification is shown for each case. Bottom, left: continuous and categorical covariates can be described via multiple options, able to define, where applicable, their distribution, interpolation, transformation, or declaration (for new covariates). Bottom, right: the variability model keeps the available information about parameter- and residual-error-related variability levels and their structure.

Examples") providing for each example the PharmML code, the corresponding codes for execution in at least one target software, and the output files obtained from the executions. The first use case describes the PK/PD of the anticoagulant drug warfarin; we use it to demonstrate typical PK/PD workflows including the estimation of parameters using data from a clinical study, the introduction of interoccasion variability on the PK parameters, the modeling of side effects through time-to-event models, and the use of the estimated parameters to design a new study through optimal design. The second use case depicts the minimal model of glucose kinetics and insulin action and shows how to define a single-subject model and an explicit trial design. Finally, the third use case demonstrates the connection between PharmML and

SBML by reimplementing the mitotic oscillator model from the BioModels repository.[14,15,43] Moreover, the **"Supplementary information – Code snippets"** file shows additional useful code snippets destined to highlight further capacities of this new language and to illustrate the concepts presented within this tutorial.

MODEL DEFINITION

The Model Definition is the core of PharmML. It allows to encode all required components of NLME models, used to describe studies in which data are collected repeatedly (usually at different timepoints) from several experimental units (subjects). Indeed, the language contains elements to declare the models for the variability structure, the covariates, the parameters, the (deterministic) structure (of the predicted data), and the observations, as shown in **Figure 2** (left). The theory of NLME is beyond the scope of the present article and can be found for instance in Lavielle's book[6] or Bonate's book,[5] from which we borrow mathematical notations and definitions here in order to explain the models covered by and expressed in PharmML. Following Lavielle,[6] we define a model as a joint probability distribution describing the relationships between variables. Assuming i denotes the subject index and y_{ij} is the j^{th} observation in this subject, a general NLME model is shown in **Figure 3**. The first description level is the observation model, describing the probability distribution of y_{ij} for given values of some independent variables x_{ij} which may be controlled in the experiment (also called design variables, such as doses and time, t, in a PK study), conditional to the subject-specific set of parameters for the subject i, ψ_i. For continuous responses, the ψ_i are the parameters of a structural model, f. For discrete responses, the observation model may require defining specific elements, for example the categories (*cat1*, *cat2*, ..., *catK*) of a categorical outcome with their associated probabilities (π_1, π_2, ..., π_K), or the survival (S) or hazard (h) function associated with an event, the outcome of a time-to-event model. The second description level of a general NLME model is the parameter model, describing the distribution of the parameters ψ_i: each individual parameter is described through a typical population parameter, ψ_{pop}, some individual covariates, c_i, the coefficients of the fixed effects from those covariates, β, and a subject-specific random effect, η. Both the continuous responses of the observation model and the individual parameters of the parameter model can require transformations ($u(\cdot)$ in the former and $h(\cdot)$ in the latter case): as detailed below, PharmML allows to encode the transformed (or untransformed) data and parameters in multiple ways, i.e., through the definition of distributions, structured equations, or arbitrary functions ($U(\cdot)$ in the observation model and $H(\cdot)$ in the parameter model).

Various elements of the NLME models, e.g., the probability of observing a given residual error, or the distribution of parameters or covariates, are described using a wide range of probability distributions. Encoding of these elements can be greatly facilitated with an external reference database of distributions. For this purpose, DDMoRe developed ProbOnto (http://probonto.org), a knowledge base and ontology covering a large family of distributions.[26] ProbOnto now contains over 100 univariate and multivariate probability distributions along with their defining functions and interrelationships, and has proved to be particularly helpful in the encoding of discrete models, abundant in pharmacometrics, such as the zero-inflated Poisson, zero-inflated negative binomial, and generalized Poisson models, with various parameterizations.[22–24,27]

The rest of this section describes the different parts of the PharmML Model Definition. A detailed overview of its structure is provided by **Figure S2** in the "**Supplementary information – Figures**" file.

Observation model

PharmML provides comprehensive support for the encoding of observation models for continuous and discrete data via the `<ObservationModel>` block.

Continuous case. A large number of observation models can be written as a structured equation corresponding to the following template:

$$u(y) = u(f) + g(\ldots)\,\epsilon \qquad (1)$$

where u defines a transformation, such as log, logit, Box-Cox, and probit, ε is a residual error with symmetrical distribution with mean 0, and $g(\ldots)$ expresses the standard deviation of the residual variability, which may depend on the prediction f and on additional parameters. The default identity transformation, $u(y) = y$, is equivalent to an untransformed data model. This is called the structured observation model. It is encoded within the `<Standard>` element, and is able to handle untransformed and both-sides-transformed data, and reads in (abbreviated) PharmML as:

```
<ObservationModel blkId=om1>
    <Transformation type=u/>
    <ContinuousData>
        ...
        <RandomVariable>...</RandomVariable>     <!- epsilon definition ->
        <Standard symbId=y>
            <Output>...</Output>                 <!- reference to f ->
            <ErrorModel>...</ErrorModel>         <!- definition of g ->
            <ResidualError>...</ResidualError>   <!- reference to epsilon ->
        </Standard>
    </ContinuousData>
```

</ObservationModel>

In this and other code snippets we use attributes, such as *symbId/symbIdRef, blkId/blkIdRef, oid/oidRef*, which help to declare/define and reference model/trial design elements, and comments. See **Chapter 1** in the "**Supplementary information – Code snippets**" file (referred to in the following in the abbreviated form **CS-1**) for a detailed explanation of their usage.

A special case of a structured observation model is the Gaussian model, by far the most commonly used in NLME models, with $\varepsilon \sim N(0,1)$. This means that $u(y)$ in (1) is normally distributed with mean $u(f)$ and standard deviation $g(\ldots)$. The Gaussian implementation for the combined error model

$$y = f + (a + b\,f)\,\epsilon \qquad \epsilon \sim N(0,1) \tag{2}$$

where a and b are the parameters of the error model, is provided in **CS-2.1**.

Another option to specify the observation model is the explicit distribution notation, used for example in WinBUGS, which in its general form reads

$$u(y) \sim DistribName(p_1, p_2, \ldots) \tag{3}$$

where *DistribName* is the name of a suitable distribution and *p1*, *p2*, ... are its parameters. (2) reads in this format

$$y \sim N(f,\ (a + b\,f)^2) \tag{4}$$

which corresponds to the PharmML code in **CS-2.2**, using the element <General> and its child element <Distribution>.

A third option, using the <AssignStatement> element, is to utilize an explicit equation-type notation for which an arbitrary expression on the right-hand side is permitted:

$$u(y) = U(f, \xi, \epsilon_1, \epsilon_2, \ldots) \tag{5}$$

where U is an arbitrary function of its arguments, i.e., the structural model, f, the parameters, ξ, and any number of residual errors defined as normally distributed random variables, ε_1, ε_2, ... Some models can only be expressed using this last notation: an example is provided in **CS-2.3**.

Discrete case. Noncontinuous data can be categorized into categorical, count, or time-to-event data, and all of them can be implemented via the <Discrete> element. Within the <CategoricalData> subelement, nominal categorical data models are typically specified by assigning probabilities to each category, i.e., defining the probability mass function of the Bernoulli or categorical distribution. For ordered categorical data, e.g., pain score records, the model formulation involves (transformed) cumulative or tail probabilities. For example, the proportional odds model[28] may read

$$logit(P(y_{ij} \le cat1)) = \theta_1 + \theta_2\, f(x_{ij}, \psi_i) \tag{6}$$

where P is the probability operator, *cat1* is a data category, and Θ_1 and Θ_2 are model parameters. The corresponding proportional odds model reads in PharmML as specified in **CS-2.4**.

Count or frequency data models, described via the <CountData> subelement, specify an appropriate distribution,[22,24] e.g., Poisson, and a relationship between the distribution parameters (λ in the following example) and the explanatory variables (e.g., the drug concentration):

$$y_{ij} \sim Poisson(\lambda(x_{ij}, \psi_i)) \tag{7}$$

The matching code snippet is shown in **CS-2.5**.

In time-to-event analyses, the random variable represents the time until the occurrence of an event. The model, described by the <TimeToEventData> subelement, is fully defined by a survival or a hazard function (see **CS-2.6** for an example).

Markov dependencies are very common in NLME models and can also be expressed in PharmML (see **CS-2.7** for an example).

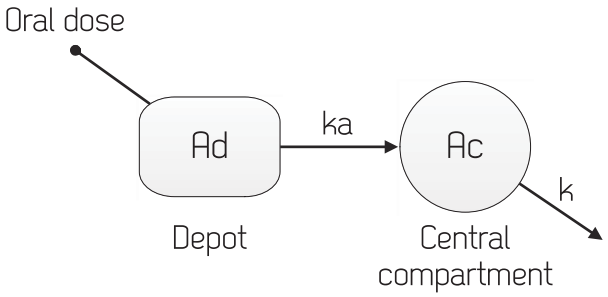

Figure 4 Schematic representation of a PK model for oral administration to a one-compartment distribution model with linear elimination. *Ad* and *Ac* are the drug amounts in the depot and central compartments, respectively; *ka* is the first-order absorption rate from the depot compartment and *k* is the first-order elimination rate from the central compartment. Virtually any compartmental PK model can be represented alternatively by a set of ODEs, algebraic equations, or PK macros.

Structural model

Within pharmacometric analyses, it is commonly assumed that the same structural model f can be used to describe different subjects in the population, while the model individual parameters express the between-subject variability. The structural model is defined as a parametric function, and PharmML offers multiple options to express it within the <StructuralModel> block. Using a basic oral PK model as an example (**Figure 4**), we describe here three of them: ordinary differential equations (ODE), algebraic functions, and PK macros.

The following system of equations provides the complete mathematical formulation of the considered PK model:

$$\frac{dA_d}{dt} = -k_a\, A_d \tag{8}$$

$$\frac{dA_c}{dt} = k_a\, A_d - k\, A_c \tag{9}$$

$$C = \frac{A_c}{V} \tag{10}$$

where t is the time variable, A_d and A_c are the drug amounts in the depot and central compartments, respectively, C is the drug concentration in the central

compartment, with volume V, k_a is the first-order absorption rate from the depot compartment, and k is the first-order elimination rate from the central compartment. Assuming a single dose administration, D, at initial time 0 leads to the initial conditions $A_d(t=0) = D$ and $A_c(t=0) = 0$. The two ordinary differential equations can be implemented via the `<DerivativeVariable>` element, describing the right-hand term of the ODE and its initial condition (see **CS-3.1** for the code for (8)).

An ODE system can contain any number of equations, meaning, for example, that complex physiology-based pharmacokinetics (PBPK) models are encodable as well.

The equations system above has an algebraic solution that can be alternatively used in PharmML directly via the `<Variable>` element (see the definition of C in **CS-3.2** as an example of variable assignment).

$$A_c(t) = \frac{k}{k_a - k}\left(e^{-kt} - e^{-k_a t}\right)$$

PharmML supports yet another way to express this PK model, in the form of the PK macros used in MLXTRAN,[29] the language of Monolix.[10] This is a powerful way to formulate compartmental PK models from their schematic representation, in this case:

```
compartment(amount=Ac, concentration=C, volume=V)
oral(ka)
elimination(k)
```

Two PharmML examples using the element `<PKmacros>` are provided in **CS-3.2**.

In addition, several structural models can be combined together in PharmML according to the modularity principle. Finally, PharmML also offers the possibility to define models involving delayed differential equations.

Parameter model

The `<ParameterModel>` block is the place where typical parameter values, fixed effects of covariates, random effects and their correlation can be integrated into an individual parameter model, described via the `<IndividualParameter>` element. Individual parameters are defined as the combination of fixed effects, which are shared across subjects and represent the typical values of the parameters in a group of individuals with the same covariates values, and random effects, which characterize the single subject. The latter are introduced via the `<RandomVariable>` element.

Three equivalent models can be used to express a parameter model and are described below. **CS-4.1** to **CS-4.4** provide the code snippets for their equations and for an example of mixture model.

Structured model (S-type). The characteristic of this representation is that all its elements are unambiguously defined through specialized PharmML elements. A typical S-type model consists of a transformation, a typical value, the fixed effects of the covariates, and the random effects.[6] The following equation, for example, describes a simple lognormally distributed volume parameter, with individual value V_i, typical value V_{pop} and random effect η_V:

$$\log(V_i) = \log(V_{pop}) + \eta_V \tag{11}$$

A straightforward extension describes the so-called linear covariate model. Covariates inform the pharmacometric model about patient demographics, laboratory measurements, and many other intrinsic/extrinsic factors used to explain part of the variability in the pharmacokinetics or pharmacodynamics of a drug.[5] Using here the body weight, W, with median equal to 70 kg, and denoting the associated fixed effect with β_V, (11) can be extended to:

$$\log(V_i) = \log(V_{pop}) + \beta_V \log(W/70) + \eta_V \tag{12}$$

Parameter models with a nonlinear covariate model in which multiple fixed/random effects and covariates are incorporated are also allowed in PharmML (**CS-4.5**).

Distribution-based model (D-type). In this case, the parameter distribution is explicitly specified, without referring to the random effect. The D-type model corresponding to (12) reads:

$$\log(V_i) \sim \mathcal{N}\left(\log(V_{pop}) + \beta_V \log(W/70), \omega_V\right) \tag{13}$$

where ω_V is the standard deviation of the chosen normal distribution.

Equation-based model (E-type). This is the most flexible declaration, with the general form $\psi_i = H(\beta, c_i, \eta_i)$, where H is a transformation and β is the set of coefficients for the fixed covariate effects. Any expression and any number of fixed/random effects and covariates is allowed using this model. The lognormally distributed parameter described by (12) and (13) reads in this case:

$$V_i = V_{pop}\,(W/70)^{\beta_V}\, e^{\eta_V},\ \ \eta_V \sim N(0, \omega_V) \tag{14}$$

Parameter mixture models are supported as well, through the *MixtureDistribution* in ProbOnto.

Each representation of the parameter model can handle any number of nested variability levels, as introduced in **Variability Model**, below. For example, a study in which every subject is followed on two or more occasions (**Figure 5**) needs to be described with two levels of variability, the reference subject level denoted by index i and the occasion level denoted by index k. In this case, for a simple model on volume V, without covariates, the three equivalent representations mentioned above read:

$$\log(V_{ik}) = \log(V_{pop}) + \eta_i^{(0)} + \eta_{ik}^{(1)} \tag{15}$$

$$\log(V_{ik}) \sim \mathcal{N}\left(\log(V_{pop}), \omega + \gamma\right) \tag{16}$$

$$V_i = V_{pop}\,(W/70)^{\beta_V}\, e^{\eta_i^{(1)}}\, e^{\eta_{ik}^{(1)}} \quad \text{with } \eta_i^{(0)} \sim N(0, \omega) \text{ and } \eta_{ik}^{(1)} \sim N(0, \gamma). \tag{17}$$

The PharmML implementations for models (15) and (16) are provided in **CS-4.6**, while model (17) is a straightforward modification of (14).

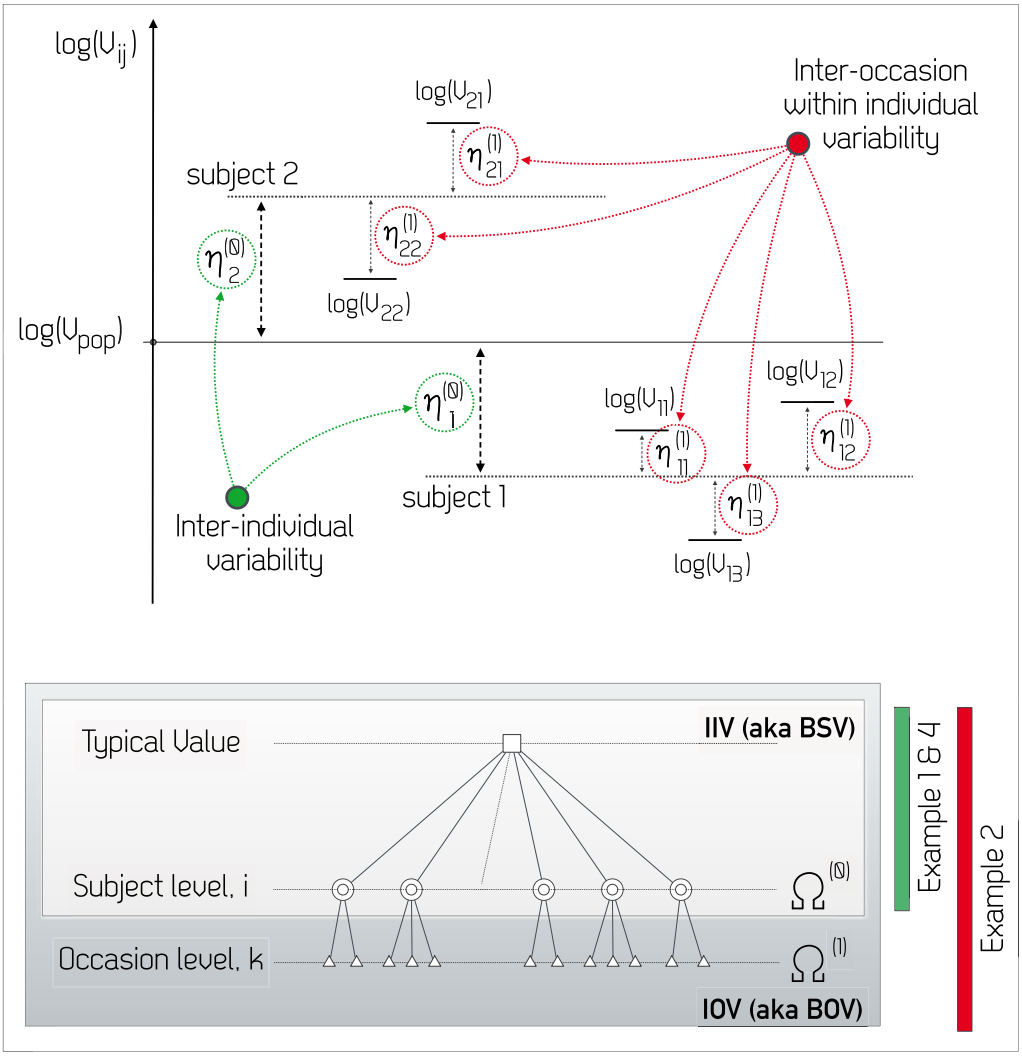

Figure 5 Parameter-related variability arising from the analysis of the study design. Two sources of unexplained parameter variability are represented: the interindividual (IIV) and interoccasion (IOV) ones. Variability affects here the volume of distribution, *V*, which is assumed to be log-normally distributed. Starting from a given typical value, log(*Vpop*), the IIV level arises first and is denoted with the superscript (0) (the top panel visualizes only two subjects), followed by the IOV level, with superscript (1) (the top panel visualizes two occasions for subject 1 and three occasions for subject 2). Some models from the "**Supplementary information – Examples**" file (bottom panel) exemplify the implementations of both IIV and IOV.

Correlation of random effects within each level of variability can be expressed in PharmML pairwise, referring to the corresponding couple of random effects, or using correlation or covariance matrices (see **CS-4.7** for an example).

Typical parameters can also be assigned a distribution, in the context of hierarchical modeling or Bayesian inference, or any arbitrary algebraic expression. Finally, basic parameters can be instantiated as well, through a simple parameter declaration or an algebraic expression assignment. This is useful in QSP, where the distinction between individual or population parameters is usually not required or possible.

Covariate model

The covariates used to define the individual parameters are described in the `<CovariateModel>` block, which provides the modelers with multiple options to declare and perform various operations on categorical and continuous covariates. The definitions in this model can be overwritten by declarations in the `<TrialDesign>` block. The available options are provided below.

Distribution. As an example, the following snippet implements a conditional log-normal distribution of body weight, *W*, for male, $SEX = M$, and female, $SEX = F$, with gender-specific parameters for the mean, $\mu_{(F,M)}$, and the standard deviation, $\sigma_{(F,M)}$:

$$W \sim LN(\mu_F, \sigma_F) \text{ if } SEX = F \tag{18}$$

$$W \sim LN(\mu_M, \sigma_M) \text{ if } SEX = M \tag{19}$$

This is the PharmML code (see **CS-5.1** for the full implementation), using the `<Distribution>` element:

```
<CovariateModel blkId="cm1">
   <Covariate symbId="W">
      <Continuous>
         <Distribution>
            <math:Piecewise>
               <!-- piecewise formula
               skipped for brevity -->
            </math:Piecewise>
         </Distribution>
      </Continuous>
   </Covariate>
</CovariateModel>
```

Assignment. In the Cockcroft–Gault model of creatinine clearance,[30] the latter, *CrCL*, is defined based on four other covariates, i.e., body weight, *W*, sex, *SEX*, age, *Age*, and serum creatinine concentration, *Scr*:

$$CrCL = \frac{(140 - Age)\, W\, 0.85}{72\, Scr} \quad \text{if } SEX = F \tag{20}$$

$$CrCL = \frac{(140 - Age)\, W}{72\, Scr} \quad \text{if } SEX = M \tag{21}$$

For categorical covariates, e.g., *Sex*, a simple declaration of the categories is fully sufficient for an estimation or a design evaluation/optimization task, while the probabilities associated to each category also need to be included for a simulation task. **CS-5.2** and **CS-5.3** provide the codes for the definitions of both creatinine clearance and sex.

Interpolation. When continuous covariates are provided through datasets or lookup tables, they appear as time-discrete values. Within the `<Interpolation>` element, the modeler can decide to use her/his own interpolation algorithm for the intermediate values or choose one from the following list: *constant*, *nearest*, *linear*, *spline*, *pchip*, *cubic*, *lastValue*. PharmML allows also to declare independent interpolation variables with respect to which the covariate of interest is to be interpolated: e.g., body weight may be interpolated with respect to age or trial duration (see **CS-5.4** for an example).

Transformation. Covariates, once defined, may be transformed through a mathematical expression, yielding a new symbol identifier (*symbID*) via the `<TransformedCovariate>` element to distinguish it from the base covariate. This is especially useful in connection with the structured parameter model (S-type, see **Parameter Model**). Instead of creating a new covariate, a desired transformed form is defined and then used within the linear covariate model of an individual parameter (see (12) for an example). A typical example is the body weight transformation, implemented in **CS-5.5**.

Random realization. For simulation purposes, assignment of randomly sampled values according to any distribution featured in ProbOnto is also available (see **CS-5.6** for an example).

Variability model

The variability structure is defined in PharmML by the `<VariabilityModel>` block. Variability can be defined for both the model parameters and the observations. **Figure 5** shows an example with two levels of variability for a parameter: interindividual variability, representing differences between subjects, and interoccasion variability, representing parameter fluctuations within the same subject at different periods (called occasions) during a study. This can be implemented in PharmML as follows:

```
<VariabilityModel blkId="vm1"
type="parameterVariability">
   <Level referenceLevel="true" symbId="indiv"/>
   <Level symbId="iov1">
      <ParentLevel>
         <ct:SymbRef symbIdRef="indiv"/>
      </ParentLevel>
   </Level>
</VariabilityModel>
```

where the first variability level, called *"indiv,"* is considered as the reference and used as the parent level for the second level of variability, called *"iov1."* A straightforward extension is, for example, the inclusion of different study centers at one level of variability above the subject level.

Finally, more than one variability level can be defined on the observations, thus allowing to model the so-called inter-replicate variability, i.e., the random differences between different replicates of the same measurement. A hierarchy between the level of the observations and the level of the replicates is defined similarly to what is done for individual parameters at different occasions.

Final notes

As discussed above, the majority of parameter and observation models is encodable in three equivalent forms, as structured, distribution-based, and equation-based models. However, whenever possible models should be encoded as structured models in PharmML, which ensure the highest degree of interoperability, as they can be automatically transformed into the other two types. The reverse direction does not work without human supervision or would require a powerful symbolic software for translation. Therefore the E-type, despite being the most flexible, is not interoperable. Moreover, some estimation algorithms, e.g., SAEM, require structured models to work efficiently.

This section concludes the description of the Model Definition components. Many different pharmacometric and QSP models can be composed by combining these elements, see **Figure 6** (top).

TRIAL DESIGN

The combination of a model and its parameter values defines a system that can be used to obtain model predictions and simulated values in response to given inputs. The Trial Design, described in **Figure 6** (middle) and **Figure S3** in the "**Supplementary information – Figures**" file, specifies these inputs, as well as when and where to sample outputs. The PharmML code for this section is given in **CS-6**.

Explicit design

For an ODE system, inputs can be the initial values for each model component (e.g., the initial amounts in the

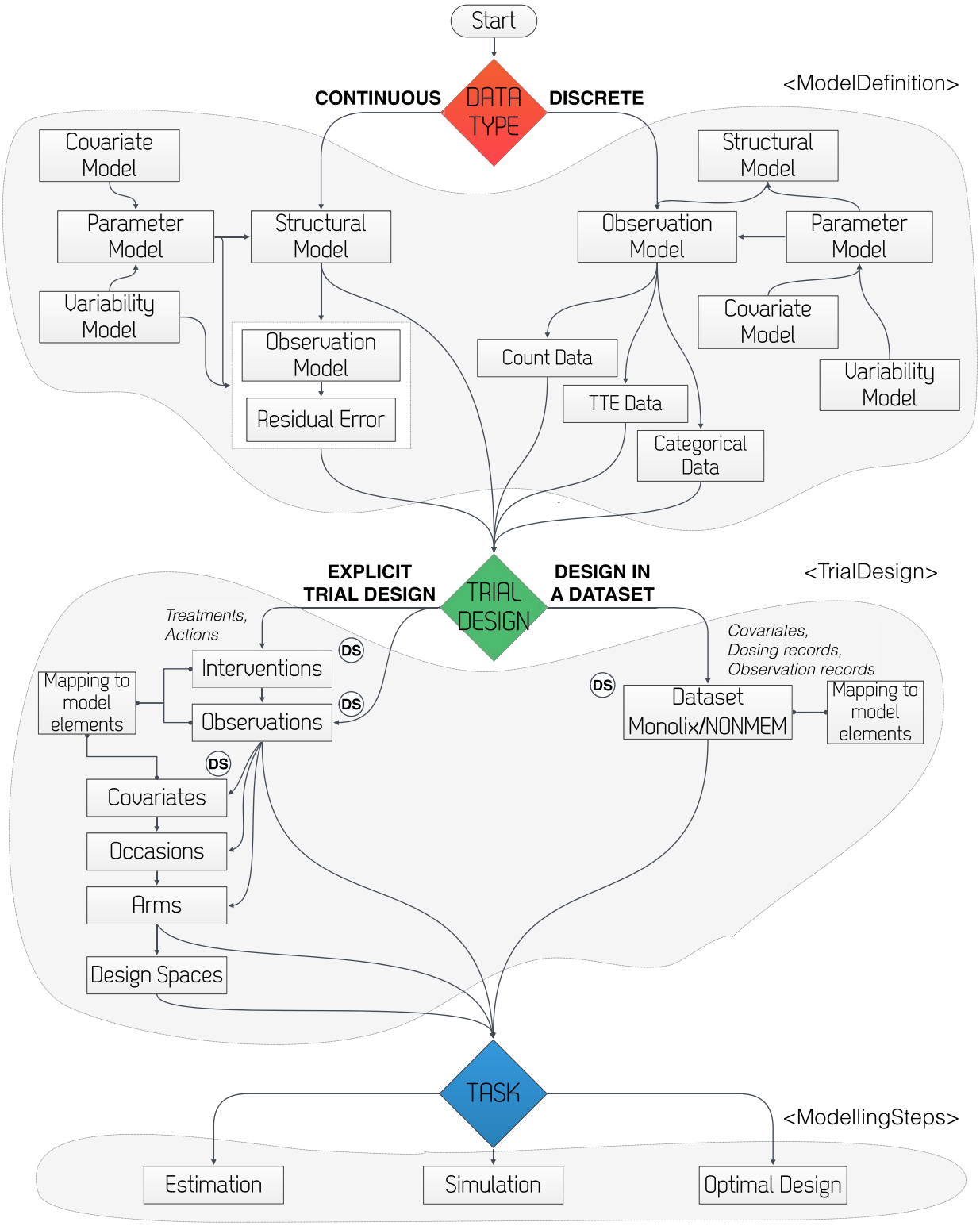

Figure 6 Model-Design-Task: the creation process of a whole PharmML model out of its different components. Top: data type determines the observation and structural model choice. Middle: trial design can be sourced from a dataset or encoded explicitly. Dependent on the clinical study characteristics, the modeler can encode a single subject design, a standard population design, or an optimal design. Bottom: once the model and design are specified, the task and the related settings remain to be declared (see **Figure 8** for a detailed representation of this block). The DS (dataset) mark identifies elements where data records can be stored inline or, alternatively, where references to external dataset files can be defined.

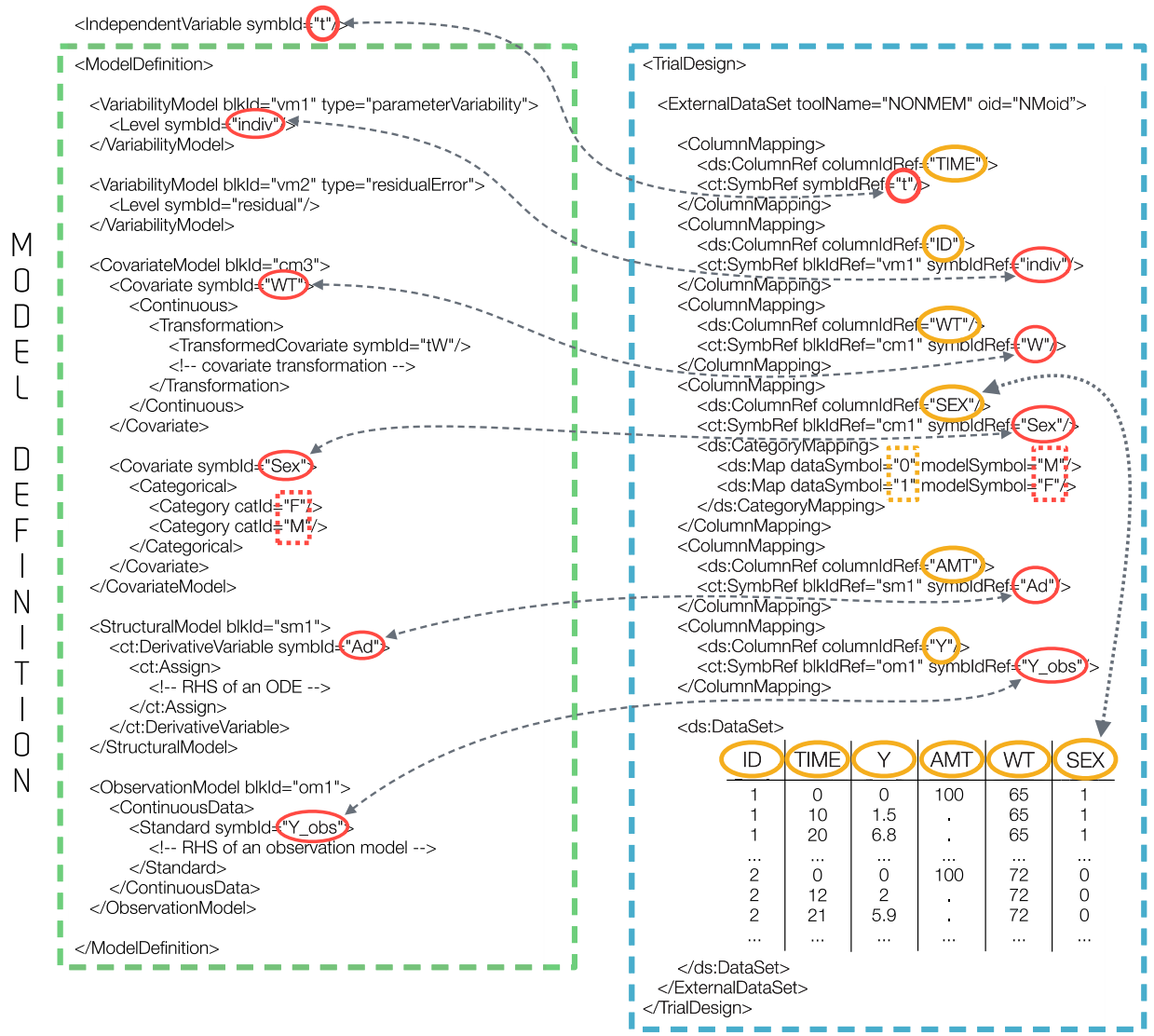

Figure 7 Code snippet exemplifying data-model mapping. Left: declaration of the independent variable and abbreviated model definition (only the main blocks are shown with indication of their content). Right: complete dataset declaration with column mapping (the dataset is shown as a table for better readability). One `<ColumnMapping>` element is used for each couple model variable (red) – correspondent data column (yellow). In certain cases, such as with categorical covariates, the `<CategoryMapping>` element is required to map the categorical symbols to the respective numerical symbols used in the dataset.

compartments) as well as subsequent changes introduced from outside the system (typically in pharmacokinetics, an additional dose can be viewed as an instant change in the value of one variable). The first component of the Trial Design is therefore the intervention(s) or treatment(s) received by the subject, which may include drug administrations but also a number of other actions (washout, variable reset, nonmedical treatments, drug-free periods, etc.). This information is enclosed in the `<Administration>` block, that can contain any number of `<Action>`, `<Bolus>`, or `<Infusion>` components (see **CS-6.1)**.

The second component of the Trial Design, `<Observations>`, defines the observations, in particular the times at which they are collected, through the `<ObservationTime>` element, as well as the responses or variables that are measured, specifying their type, for instance `<Continuous>`. Together, these two components define an individual (or elementary) design, which is applied to one experimental unit.

When trials are performed in a population of subjects, we define the population design as the set of individual designs for each subject included in the study. Generally, subjects who have the same design are grouped together, by defining treatment arms (in an `<Arm>` block) in which all subjects receive the same inputs and have the same outputs. The arms define what we call an explicit design, and correspond to the essential elements of what is typically specified in the protocol for a clinical trial. Two additional elements can be introduced in the PharmML element `<Arms>` to create more complex and realistic designs:

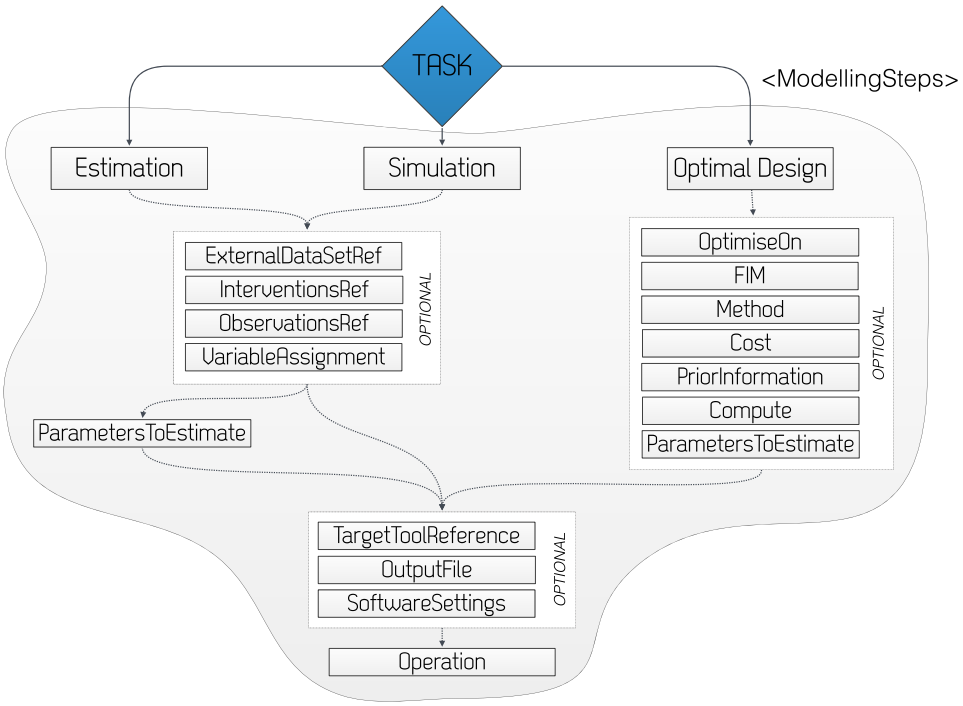

Figure 8 Modeling Steps ingredients.

1) occasions, a within-subject level of variability, may be defined with the <Occasions> element, and associated with a time frame (defined through a vector of start times) to identify different occasions for each subject in the corresponding arm; 2) covariate distributions may be defined with the <Covariates> element, and possibly supersede previous analog definitions (see **CS-6.2**).

Design spaces

There is a direct connection between the design and the information gathered during a clinical trial, which has a major impact on the precision of the parameter estimates derived in the subsequent analysis of the collected data. Design optimization aims to find the best design over possible values of certain variables, such as doses, sampling times, and number of subjects. These possible values constitute what we call a design space, in which unit spaces for a variable are defined as either a discrete set or a range of allowed values, and possibly combined to create more complex design spaces. Design spaces are defined within the block <DesignSpace> (see **CS-6.3**).

Design spaces may be defined for different elements of the Trial Design, e.g., for the <ArmSize> (the number of subjects in an arm), for user-defined variables, or for elements used within the Trial Design, like the <SampleTimes> (the times to observe a response) element of the <Observation> object.

Optimization takes place under a set of constraints including the total cost (usually equal to the number of samples, but more complex cost functions can also be used) and is executed in different software.[31] These features are specified within the Modeling Steps block of the PharmML code (see **Modeling Steps**).

Dataset declaration and data/model mapping

When a trial is actually performed, we can expect deviations from the dosing and sampling schedules specified in the protocol. For instance, a patient may forget to take a dose or make a mistake on the dose amount, or the sample collection may be delayed or omitted. In such cases, individual protocols are extracted from a dataset providing subject-specific observation times, interventions, occasions, and covariate values: the set of individual protocols constitutes the empirical population design.

PharmML has the ability to declare references to external dataset files as well as to encode dataset records explicitly, i.e., inline (see the "DS" marks in **Figure 6**, middle, and **CS-7.2** for an example). Moreover, it is possible to store all observation, dosing, and covariate data in a unique dataset (see **Figure 6**, middle right). With respect to this latter option, PharmML supports the most popular NONMEM-type dataset format or the slightly different Monolix-type format, and virtually any tabular format in which data are organized by columns. The declaration of such a dataset is located in the <TrialDesign> section, in the <ExternalDataSet> block. This block must contain a reference to the external file in which the data are stored (element <ExternalFile> in the <DataSet> block), as well as information on how to interpret each column (elements <Column> in the <Definition> component of the same block). Attributes can be used to indicate the type of data contained in a given column (continuous, categorical, etc.) and to associate keywords indicating, for instance, that the data item is to be used as a dependent variable or a covariate. Each data item is then matched to its corresponding variable in the model through the <ColumnMapping> elements. An example is given in **CS-7.1**.

A number of other options allow for more flexibility in working with datasets, such as the definition of data headers and missing data, the transformation of columns (e.g., for basic dose scaling), and the conditional mapping of columns. Further explanations and illustrative examples are provided in **CS-7.3** to **CS-7.8**.

MODELING STEPS

The <ModellingSteps> block describes the tasks typically used in pharmacometrics and QSP, such as estimation, simulation, and design evaluation or optimization (**Figure 8**). It starts with the (optional) specification of the target tool and is followed by the definition of one of three predefined elements, <EstimationStep>, <SimulationStep>, or <OptimalDesignStep>, each holding a number of specialized elements. Here we concentrate on the high-level description of the estimation task only, for which a representative complete code is available in **CS-8**.

When performing an estimation task, it is necessary to declare the data used to estimate the parameters of the model. This can be done in two ways. The first is to refer, via the <ExternalDataSetReference> element, to the NONMEM/Monolix data file defined in the Trial Design within the <ExternalDataSet> element (**CS-7.1**). The second way is to refer, via the <ObservationsReference> element, to the data stored inline within the <Observations> element defined in the Trial Design (**CS-7.2**). Moreover, the estimation process requires the initial estimates for the model parameters: they are declared within the <ParametersToEstimate> element, which has child elements for specifying the lower/upper bounds of the estimates, <LowerBound> and <UpperBound>, respectively.

One of the few mandatory elements common to all tasks is <Operation>. Here, within the <Algorithm> and <Property> child elements, the user can declare general task properties, algorithms, and their settings. It is worth mentioning at this point that target tools may require complex settings that are conveniently stored in external files (e.g., the graphical settings in Monolix[10]). Such an option is supported and each task type may contain one or more <SoftwareSettings> elements containing a reference to an external file.

Finally, each task results in numerical and/or graphical outputs, and their specification can be made within the <OutputFile> element, e.g., in the form of a Standardized Output (SO) file (see next chapter for details).

WORKING WITH PharmML

Interacting with PharmML

This tutorial has presented so far the PharmML structure and how various model elements can be implemented within it. Moreover, the online "**Supplementary information – Examples**" file contains a number of complete real-life use cases with detailed descriptions. PharmML, however, is merely an encoding format and any actual execution requires a software infrastructure to use it, i.e., to read, write, validate, and convert the XML of a PharmML model into a form understandable by a target software, and then to run the defined task.

The first three jobs are taken over by LibPharmML, a JAVA Application Programming Interface (API) also developed by DDMoRe. The API is available via SourceForge within the libPharmML project (https://sourceforge.net/projects/libpharmml.ddmore.p). A stand-alone validator, executable via command line, is also provided. Documentation for the validator is available in the libPharmML wiki (https://sourceforge.net/p/ddmore/libpharmml/wiki/Home/).

The software element that transforms PharmML to target specific code is the converter. Unlike SBML, PharmML describes both the models and the tasks performed on them. This allows a converter to read a PharmML file and to create from it executable source code for both the mathematical elements and the tasks required in the target script. The entire information necessary to transform PharmML into a target code is encapsulated within the converter; an all-encompassing software library is *not* required to execute PharmML models, as model transformation is handled by the converter and calculation elements are delegated to the target software.

DDMoRe has developed a standardized converter library for PharmML called CCoPI-Mono (CM). CM is a Java library that runs in conjunction with libPharmML to convert a PharmML model into target-specific code. CM has code generators for NONMEM, MATLAB/Octave, R, Python, C/C++, and FORTRAN. As both SBML and PharmML are interoperable, CM can convert standard SBML models into PharmML. CM has a component to convert an SBML pathway model into an ODE form, which can readily be expressed in PharmML. Once in PharmML form, an SBML-derived model can be executed in platforms that do not traditionally work with SBML, such as Monolix or NONMEM. The reverse direction, translation of PharmML to SBML, is also possible but is limited to the structural model only, as SBML does not support the encoding of any statistical model component.

Finally, if a modeler needs/wants to write a PharmML model from scratch, two DDMoRe tools may support this task. One of them is the infix2pharmml tool, available as a free web service and able to translate expressions from the usual mathematical infix notation into the corresponding PharmML markup (http://infix2pharmml.sourceforge.net/). The other tool is the MDL-IDE, a framework for creating and editing models in the human writeable and readable language MDL[32]: within the same framework, it is possible to translate the MDL code into the corresponding PharmML code. The MDL-IDE benefits from the use of the SO, another XML-based exchange format developed within DDMoRe to store the results of any pharmacometric analyses in a standardized way, so that they can be later recalled, reused, and transferred from one analysis step to another.

Use cases

The "**Supplementary information – Examples**" file provides a complete implementation of selected models illustrating the essential functionalities and capabilities of

PharmML. All models are described in detail with an explanation of the modeling background and the specific involved PharmML features. The chosen models are the following:

- Warfarin PK/PD model[33–37], in four different versions:
  1 – Basic PK & turnover PD,
  2 – Inter-occasion variability on PK parameters,
  3 – Time-To-Event model for PD data,
  4 – Optimal design application;
- Minimal model of glucose kinetics (reparameterized)[38–40];
- Mitotic oscillator.[41]

DISCUSSION

The present tutorial is an introduction to PharmML, a new markup language designed to encode models and analyses in pharmacometric and system pharmacology applications. As an XML-based language, it is structured around a series of specialized hierarchical sections, each addressing a different component of the model, starting with the Model Definition (with covariate, observation, parameter, structural, and variability submodels). The section Trial Design allows specifying the design either through a data reference or as an explicitly coded study structure. A final section, Modeling Steps, contains a description of the activities that can be associated with the model object and of the software settings and variables associated with them: typical tasks involved in pharmacometric and QSP applications are estimation, prediction/simulation, and optimal design, and all of them can be combined within PharmML to define workflows.

The PharmML language is extremely flexible both in terms of the range of models that can be encoded, and in terms of the analyses that can be set up through the task component. The "**Supplementary information – Examples**" file provides a few complete, PharmML-coded models illustrating this versatility. However, during its development, PharmML was validated across a much larger variety of examples, which helped ensure that it could be used to describe a large number of models and clinical situations. The use of SBML together with the task language SED-ML[42] was considered during the conception of PharmML, but discarded because of the intended broader scope of models and the focus on workflows. However, as shown in the examples, models coded in SBML can be easily converted and used.

PharmML deals with probability distributions through the ProbOnto module, which allows both to describe the variety of statistical models found in the definition of NLME models and to define the covariate distributions characterizing the study populations. Another strength of ProbOnto is the unified representation of probability distributions, which is consistent with the unified representation of NLME models in terms of probability distributions[6] and makes it very easy to declare noncontinuous responses. As a result, the language can describe the features of an entire clinical trial. Models may be annotated, i.e., connected either to ontologies under development in the DDMoRe community, including ProbOnto,[26] or to ontologies such as ChEBI.[43] Annotations can then be used in the search engines, extending PharmML to the semantic web. The manual available on the DDMoRe Model Repository (http://repository.ddmore.eu) provides details on the annotation process. PharmML is distributed under the Apache License, Version 2.0 license (this can be found in each schema file of the exchange format) and is available both on its website (http://www.pharmml.org/) and on the GitHub repository (https://github.com/pharmml/pharmml-spec).

There are still scenarios that might be difficult to encode in PharmML. One example could be adaptive clinical trials, for instance, when the doses are adapted based on PK or PD outcomes, or when inclusions are stopped after intermediate analyses. However, the flexibility and modularity of the language would make it easy to extend PharmML to such applications in the near future. Tasks could also be extended, for instance, with noncompartmental analyses and related target software settings.

PharmML, as part of the DDMoRe interoperability platform, or as a stand-alone exchange format, has the potential to improve the way pharmacometricians and life scientists work today in multiple ways. A model encoded once can be automatically translated into a representation used in any supported target tool. Therefore, PharmML can facilitate smooth and lossless transmission of models between tools, enable complex workflows based on standardized model definition, and improve reproducibility of research by simplifying report generation and bug tracking. If adopted across the pharmacometric field, it could also improve the interaction with regulatory agencies. As seen in the SBML community,[12] standards stimulate the development of new tools and methods and facilitate the use of existing models, as has happened with many computational models of biological processes in the BioModels database.[14,15,44] Thanks to its flexibility, modularity. and broad scope, PharmML may serve the same purpose in the Modeling and Simulation community.

Acknowledgments. The authors thank the many colleagues across the DDMoRe project who have contributed to developing, refining, and implementing PharmML. This study received support from the Innovative Medicines Initiative Joint Undertaking under grant agreement #115156, resources of which are composed of financial contributions from the European Union's Seventh Framework Programme (FP7/2007–2013) and EFPIA companies' in kind contribution. The DDMoRe project is also financially supported by contributions from Academic and SME partners.

Conflict of Interest. The authors declared no conflict of interest.

Author Contributions. R.B. and E.C. contributed equally to this work. R.B., E.C., and M.J.S. designed and wrote the manuscript, and produced the use cases. G.S. wrote the manuscript and produced the use cases. F.Y. wrote the manuscript. N.R.K. revised the manuscript.

1. Swat, M. *et al.* Pharmacometrics markup language (PharmML): opening new perspectives for model exchange in drug development. *CPT Pharmacomet. Syst. Pharmacol.* **4**, 316–319 (2015).
2. Fallside, D. C. XML Schema Part 0: Primer. <https://www.w3.org/TR/2000/CR-xmlschema-0-20001024/primer.html>.
3. Thompson, H.S., Beech, D., Maloney, M. & Mendelsohn, N. XML Schema Part 1: Structures. <https://www.w3.org/TR/2000/CR-xmlschema-1-20001024/>.
4. Biron, P.V. & Malhotra, A. XML Schema Part 2: Datatypes. <https://www.w3.org/TR/2000/CR-xmlschema-2–20001024/>.

5. Bonate, P.L. *Pharmacokinetic-Pharmacodynamic Modeling and Simulation* (Springer US, 2011).
6. Lavielle, M. *Mixed Effects Models for the Population Approach: Models, Tasks, Methods and Tools* (CRC Press, Boca Raton, FL; 2014).
7. Ette, E.I. & Williams, P.J. *Pharmacometrics: The Science of Quantitative Pharmacology* (Wiley, Hoboken, NJ; 2007).
8. Sorger, P.K. Quantitative and systems pharmacology in the post-genomic era: New approaches to discovering drugs and understanding therapeutic mechanisms. In *NIH White Pap. QSP Workshop Group* 1–48 (An NIH white paper by the QSP workshop group, 2011).
9. Beal, S., Sheiner, L.B. & Boeckmann, A. *NONMEM User's Guides (1989–2006)* (Icon Development Solutions, Ellicott City, MD; 2006).
10. Lixoft MONOLIX. (2016). <http://lixoft.com/products/monolix/>.
11. Jamei, M. *et al.* The Simcyp population based simulator: architecture, implementation, and quality assurance. *Silico Pharmacol.* **1**, 9 (2013).
12. Hucka, M. *et al.* The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinforma. Oxf. Engl.* **19**, 524–531 (2003).
13. Sauro, H.M. & Bergmann, F.T. Standards and ontologies in computational systems biology. *Essays Biochem.* **45**, 211–222 (2008).
14. Le Novère, N. *et al.* BioModels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Res.* 34, D689–691 (2006).
15. Li, C. *et al.* BioModels database: an enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Syst. Biol.* **4**, 92 (2010).
16. Olivier, B.G. & Snoep, J. L. Web-based kinetic modeling using JWS Online. *Bioinforma. Oxf. Engl.* **20**, 2143–2144 (2004).
17. Klipp, E. *et al. Systems Biology* (Wiley-Blackwell, Hoboken, NJ; 2013).
18. Harnisch, L., Matthews, I., Chard, J. & Karlsson, M.O. Drug and disease model resources: A consortium to create standards and tools to enhance model-based drug development. *CPT Pharmacometrics Syst. Pharmacol.* **2**, e34 (2013).
19. R Core Team *R: A language and environment for statistical computing* (R Foundation for Statistical Computing, Vienna, Austria; 2016). <http://www.R-project.org/>.
20. Lunn, D. J., Thomas, A., Best, N. & Spiegelhalter, D. WinBUGS — a Bayesian modeling framework: concepts, structure, and extensibility. *Stat. Comput.* **10**, 325–337
21. Mould, D.R. & Upton, R.N. Basic concepts in population modeling, simulation, and model-based drug development. *CPT Pharmacometrics Syst. Pharmacol.* **1**, e6 (2012).
22. Plan, E.L. Modeling and simulation of count data. *CPT Pharmacometrics Syst. Pharmacol.* **3**, e129 (2014).
23. Paule, I., Girard, P., Freyer, G. & Tod, M. Pharmacodynamic models for discrete data. *Clin. Pharmacokinet.* **51**, 767–786 (2012).
24. Trocóniz, I.F., Plan, E.L., Miller, R. & Karlsson, M.O. Modelling overdispersion and Markovian features in count data. *J. Pharmacokinet. Pharmacodyn.* **36**, 461–477 (2009).
25. Mould, D.R. & Upton, R.N. Basic concepts in population modeling, simulation, and model-based drug development—part 2: introduction to pharmacokinetic modeling methods. *CPT Pharmacometrics Syst. Pharmacol.* **2**, e38 (2013).
26. Swat, M.J., Grenon, P. & Wimalaratne, S. ProbOnto: ontology and knowledge base of probability distributions. *Bioinformatics* **32**, 2719–2721 (2016).
27. Chow, N. & Steenhard, D. SAS global forum: atatistics and data analysis. In *Flex. Count Data Regres. Model Using SAS® PROC NLMIXED* **250**, 1–14
28. Dobson, A.J. *An Introduction to Generalized Linear Models* (Chapman & Hall/CRC, New York; 2001).
29. Lixoft & INRIA MLXTRAN The model coding language for Monolix (2014). <http://http://mlxtran.lixoft.com/mlxtran-user-guide/>.
30. Cockcroft, D.W. & Gault, M.H. Prediction of creatinine clearance from serum creatinine. *Nephron* **16**, 31–41 (1976).
31. Nyberg, J. *et al.* Methods and software tools for design evaluation in population pharmacokinetics-pharmacodynamics studies. *Br. J. Clin. Pharmacol.* **79**, 6–17 (2015).
32. Kokash, N., Moodie, S.L., Smith, M.K. & Holford, N. Implementing a domain-specific language for model-based drug development. *Procedia Comput. Sci.* **63**, 308–316 (2015).
33. O'Reilly, R.A., Aggeler, P.M. & Leong, L.S. Studies on the coumarin anticoagulant drugs: the pharmacodynamics of warfarin in man. *J. Clin. Invest.* **42**, 1542–1551 (1963).
34. O'Reilly, R.A. & Aggeler, P.M. Studies on coumarin anticoagulant drugs. Initiation of warfarin therapy without a loading dose. *Circulation* **38**, 169–177 (1968).
35. Holford, N.H. Clinical pharmacokinetics and pharmacodynamics of warfarin. Understanding the dose-effect relationship. *Clin. Pharmacokinet.* **11**, 483–504 (1986).
36. Nagashima, R., O'Reilly, R.A. & Levy, G. Kinetics of pharmacologic effects in man: the anticoagulant action of warfarin. *Clin. Pharmacol. Ther.* **10**, 22–35 (1969).
37. Abbrecht, P.H., O'Leary, T.J. & Behrendt, D.M. Evaluation of a computer-assisted method for individualized anticoagulation: retrospective and prospective studies with a pharmacodynamic model. *Clin. Pharmacol. Ther.* **32**, 129–136 (1982).
38. Bergman, R.N., Ider, Y.Z., Bowden, C.R. & Cobelli, C. Quantitative estimation of insulin sensitivity. *Am. J. Physiol.-Endocrinol. Metab.* **236**, E667–E677 (1979).
39. Bergman, R.N., Phillips, L.S. & Cobelli, C. Physiologic evaluation of factors controlling glucose tolerance in man: measurement of insulin sensitivity and beta-cell glucose sensitivity from the response to intravenous glucose. *J. Clin. Invest.* **68**, 1456–1467 (1981).
40. Mari, A. Assessment of insulin sensitivity with minimal model: role of model assumptions. *Am. J. Physiol. Endocrinol. Metab.* **272**, E925–E934 (1997).
41. Goldbeter, A. A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase. *Proc. Natl. Acad. Sci. U. S. A.* **88**, 9107–9111 (1991).
42. Waltemath, D. *et al.* Reproducible computational biology experiments with SED-ML—the Simulation Experiment Description Markup Language. *BMC Syst. Biol.* **5**, 198 (2011).
43. Hastings, J. *et al.* The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Nucleic Acids Res.* **41** (D1), D456–D463 (2013).
44. Juty, N. *et al.* BioModels: content, features, functionality, and use. *CPT Pharmacometrics Syst. Pharmacol.* **4**, e3 (2015).

Supplementary information accompanies this paper on the *CPT: Pharmacometrics & Systems Pharmacology* website (http://psp-journal.com)