ICT 269978

Integrated Project of the 7th Framework Programme

COOPERATION, THEME 3
Information & Communication Technologies
ICT-2009.5.3, Virtual Physiological Human

VPH-Share

Work Package: WP5

Patient-Centred Multiscale Computational Workflows

Deliverable: D5.4

Mark-up Language Representation of Variation and Uncertainty

Version: 1v1

Date: 23-Feb-13

DOCUMENT INFORMATION

IST Project Num	FP7 – ICT - 269978	**Acronym**	VPH-Share
Full title	Virtual Physiological Human: Sharing for Healthcare – A Research Environment		
Project URL	http://www.vph-share.eu		
EU Project officer	Robert Begier		

Work package	**Number**	5	**Title**	Patient-Centred Multiscale Computational Workflows
Deliverable	**Number**	5.4	**Title**	Mark-up Language Representation of Variation and Uncertainty

Date of delivery	**Contractual**	28-Feb-13	**Actual**	28-Feb-13
Status	Version 1v1		Final ☐	
Nature	Prototype ☒ Report ☐ Dissemination ☐ Other ☐			
Dissemination Level	Public (PU) ☒ Consortium (CO) ☐	Restricted to other Programme Participants (PP) ☐ Restricted to specified group (RE) ☐		

Authors (Partner)	Andrew Miller (UOA), Randall Britten (UOA)			
Responsible Author	Andrew Miller		**Email**	ak.miller@auckland.ac.nz
	Partner	UOA	**Phone**	+6499235122

Abstract (for dissemination)	As a result of the work done to date on this project, new proposed standards for quantifying the uncertainty in simulation parameters and fields have been put forward, existing tools have been extended to support these proposed standards, and mathematical models and field descriptions including uncertainty have been created to demonstrate the proposed standards and tools.
Keywords	Uncertainty, Parameter Variation, CellML, FieldML, Declarative Languages

Version Log			
Issue Date	**Version**	**Author**	**Change**
07/12/2012	0.1	Andrew Miller	Chapter Headings
28/01/2013	0.4	Andrew Miller	First Draft
18/02/2013	1.0	Andrew Miller	Internal Review Complete (Nic Smith & Susheel Varma)
23/02/2013	1.1	PMO	Submission Version

Contents

LIST OF FIGURES

EXECUTIVE SUMMARY

This report discusses how mathematical models and field descriptions represented in VPH mark-up languages can include descriptions of parameter uncertainty.

It aims to:

1. Describe the benefits that have already been captured from the task, as well as a description of benefits, which are expected in future collaborations.
2. State the measurable outcomes to date from the tasks.
3. Present the technical products that have resulted from the tasks and how they can be used by other VPH-Share partners and the broader scientific community to encode parameter uncertainty information.
4. Discuss future work which is required using the remaining allocated effort to maximise capture of benefits from the task.

The report discusses how proposed extensions have been put forward for both CellML and FieldML, both as an incremental change to the existing versions and as part of larger changes to improve the specifications for the future, how these proposed specification changes have been implemented in extensions to existing software tools, and how exemplar models containing parameter uncertainty have been created. An overview of the specification and a demonstration of how the tools developed to date can be used are included.

It additionally discusses how further work is required to create a more powerful version of FieldML that can describe parameter uncertainty in terms of a probability density function.

1 INTRODUCTION

This report discusses how mathematical models and field descriptions represented in VPH mark-up languages can include descriptions of parameter uncertainty.

The report is intended to provide information useful to several distinct groups of readers:

1. Those interested in the status of the task to extend model and field mark-up languages.

2. Those seeking an understanding of how the effort for this part of the VPH-Share project is being used.

3. Those interested in adopting the VPH mark-up languages to apply to other tasks in the VPH-Share project and to other projects.

2 BENEFITS FROM THE TASK

A number of benefits have already been captured as a result of the work done to date:

1. A model of uncertainty in cardiac shape encoded in FieldML has been made available in the Anatomical Models Database, providing a benefit of greater dissemination of cardiac shape uncertainty data encoded in a standard format.

2. Software that supports the CellML parameter uncertainty proposed standard is now in use by a significant number of people and research groups. At the date of writing, there have been 1068 downloads of versions of the CellML API that support parameter uncertainty, suggesting that the uncertainty mark-up can be accepted by a significant number of groups.

However, there are significantly more benefits which have not yet been captured, but which can be expected in the future:

1. More widespread adoption of the proposed standard will mean that mathematical model exchange within the VPH project will allow uncertainty information to be exchanged reliably and efficiently between research groups, promoting increased understanding of the uncertainty associated with mathematical models and decreased costs associated with exchanging parameter uncertainty information.

2. The efforts to improve the CellML and FieldML representations of uncertainty under this project have necessitated more general improvements to the CellML and FieldML standards. These more general improvements additionally allow a much wider range of models to be represented in the respective languages, which can be expected to improve the ability to exchange a diverse range of mathematical models between software tools and research groups in the future.

3 PRODUCTS AND MEASURABLE OUTCOMES FROM THE TASK

3.1 A proposal on describing parameter uncertainty in CellML 1.0 / 1.1

3.1.1 Proposed standard

CellML 1.0 and CellML 1.1 are established standards, which were finalised before the commencement of the VPH-Share project. Because they are in widespread use, and are supported by existing tools, it is important to have a way to describe parameter uncertainty with only a minor incremental change to these standards, rather than only providing a solution that is usable with the next version of CellML.

A proposal for such an incremental extension was published by Miller *et al.* [1], and a brief summary of the proposal is given here.

CellML 1.0 and CellML 1.1 make use of Content MathML 2 to describe the mathematical constraints that apply to mathematical models. Content MathML 2 allows for additional operators beyond those defined in the MathML specification to be referenced using a mechanism known as csymbol. Using this mechanism, three new operators were defined, allowing parameter uncertainty to be specified in CellML.

The uncertainParameterWithDistribution[1] operator takes two arguments; the first is either a variable or a vector of variables, and specifies the variable (or vector of variables) representing the parameter with uncertainty. The second argument must be a parameter distribution, described by one of the remaining new operators. This operator corresponds to the 'distributed as' operator commonly represented using the ~ symbol when equations are graphically rendered. When a parameter distribution is described in this way, it should be considered a posterior distribution; i.e. as a statement by the modeller that, given their prior beliefs and all evidence available to them, the specified distribution reflects their current belief about the probability distribution for the true value of the uncertain parameter in a given particular instance of the model (for example, in a given individual).

The distributionFromDensity[2] operator is used to define a distribution from a probability density function (*pdf*). The *pdf* is specified using a MathML lambda expression. Because CellML 1.0 and CellML 1.1 do not include vector mathematics facilities, the variable the *pdf* is described over must be a scalar, which limits the expression to a univariate *pdf*. However, the *pdf* can refer to other variables in the model (including other uncertain parameters), which means that multivariate probability distributions may be specified. Instead of $(v_1, v_2, .., v_n) \sim$ SomeMultivariateDistribution, the system can be described as:

$v_1 \sim$ MarginalDistribution

$v_2 \sim$ ConditionalDistribution$_2(v_1)$

...

$v_n \sim$ ConditionalDistribution$_n(v_1, ..., v_{n-1})$

[1] http://www.cellml.org/uncertainty-1#uncertainParameterWithDistribution

[2] http://www.cellml.org/uncertainty-1#distributionFromDensity

The operator distributionFromMass[3] works in an analogous way to distributionFromDensity[4] operator, except that it takes a probability mass function to describe an uncertain parameter with a discrete distribution, rather than one with a continuous distribution.

The above two operators are useful in the case where the closed form of the distribution of the parameter is known. However, several popular ways to produce parameter distributions do not result in closed forms for the distribution of parameters. For example, the distribution for a population might be known only by a large number of individual values from that population, or a posterior distribution might be computed as a series of samples from a Gibbs Sampling Bayesian Inference tool like WinBUGS, and have no closed form as a *pdf*. The operator distributionFromRealisations[5] allows for a distribution to be specified as a series of realisations of the distribution (i.e. samples from that distribution). The argument to the operator may either be a vector of scalar realisations (for the univariate case), or a vector of realisation vectors (for the multivariate case).

In addition to the proposed changes to CellML, a proposed change to SED-ML (the Simulation Experiment Description Markup Language) was put forward. SED-ML describes simulation experiments, by describing details of numerical algorithms and parameters to use. Simulation experiment descriptions are made up of tasks; the only task available in SED-ML Level 1 Version 1 is 'Uniform Time Course'. The proposed addition adds an additional task type called repeatedAnalysis, which behaves like UniformTimeCourse, but adds an additional attribute, numberOfSamples, giving the number of times to repeat the simulation with newly sampled values of uncertain parameters. This allows for Monte Carlo sensitivity analysis simulation experiments to be described.

3.1.2 Interoperability between the proposed standard and UncertML

UncertML is existing specification for describing information about uncertainty. It allows for three types of uncertainty information to be provided: statistics (such as mean, variance, and range), distributions (as a selection from a controlled vocabulary of choices), and samples (as a collection of realisations). UncertML was considered and rejected for use in the Physiome standards proposals presented here because it does not fit naturally with existing base standards like MathML that are in use, and because it prescribes a controlled vocabulary of options for distributions, it would be less general than the rest of CellML.

However, to show that interoperability is possible between the proposed standard for CellML parameter uncertainty and UncertML, a program that converts UncertML to MathML for inclusion in CellML models was created, and similarly a program that converts *pdf*s and *pmf*s (probability mass functions) for recognised distributions from the MathML in CellML models to UncertML was created. Multivariate distributions in UncertML are converted to marginal and conditional univariate distributions in MathML. There is more than one way to write the *pdf* for many distributions, but the converter from MathML to UncertML is limited

[3] http://www.cellml.org/uncertainty-1#distributionFromMass
[4] http://www.cellml.org/uncertainty-1#distributionFromDensity
[5] http://www.cellml.org/uncertainty-1#distributionFromRealisations

to ways of expressing the *pdf* that it is able to recognise (which are sufficient to reverse the translation from UncertML to MathML). The source code for this program is publicly disseminated at https://github.com/A1kmm/uncertml_to_physiome.

3.1.3 Support for the proposed standard in existing tools

The proposed standard was implemented in the CellML API (application programming interface) Implementation. The CellML API Implementation is a library, which provides services those developers of tools like CellML model editors, and simulators can use to easily support CellML. One of the services provided by the CellML API is the Code Generation Service, which can translate a CellML model into imperative code that will solve the model. Another service is the CellML Integration Service, which solves (numerically integrates) models describing differential-algebraic equation initial value (DAE-IV) problems.

The CellML Code Generation Service and CellML Integration Service have been extended so that they will numerically sample from models with uncertain parameters described using a *pdf* or as realisations. With this implementation, it is possible to perform a Monte Carlo analysis of the distribution of model outputs by repeatedly running a simulation with the CellML Integration Service.

This change is available in version 1.12 of the CellML API, and can be downloaded at http://cellml-api.sourceforge.net/download.html#1.12.

3.1.4 An example CellML model with uncertain parameters

To demonstrate the proposed standard (and to test tools), a number of example models were created. Some of those were added to the set of test cases used for the CellML API.

As an example of uncertainty added to a real biological model, the well-known 1977 ventricular myocardial fibre model by Beeler and Reuter was modified to add uncertainty to the parameter x1_open. The resulting model can then be simulated to determine the effect of the parameter uncertainty on other simulation results (such as the membrane voltage at a particular time). Alongside the example, a SED-ML description of a simulation experiment to perform a sampling sensitivity analysis was also created. The model and SED-ML description are publicly available at CellML Model Repository. [6]

The results of using the CellML API to simulate this model are shown below (Figure 1); the second graph omits some points to show the cluster of results around V=-84.617 mV in more detail (Figure 2).

[6] http://models.cellml.org/w/miller/beeler_reuter_1977_uncertexample

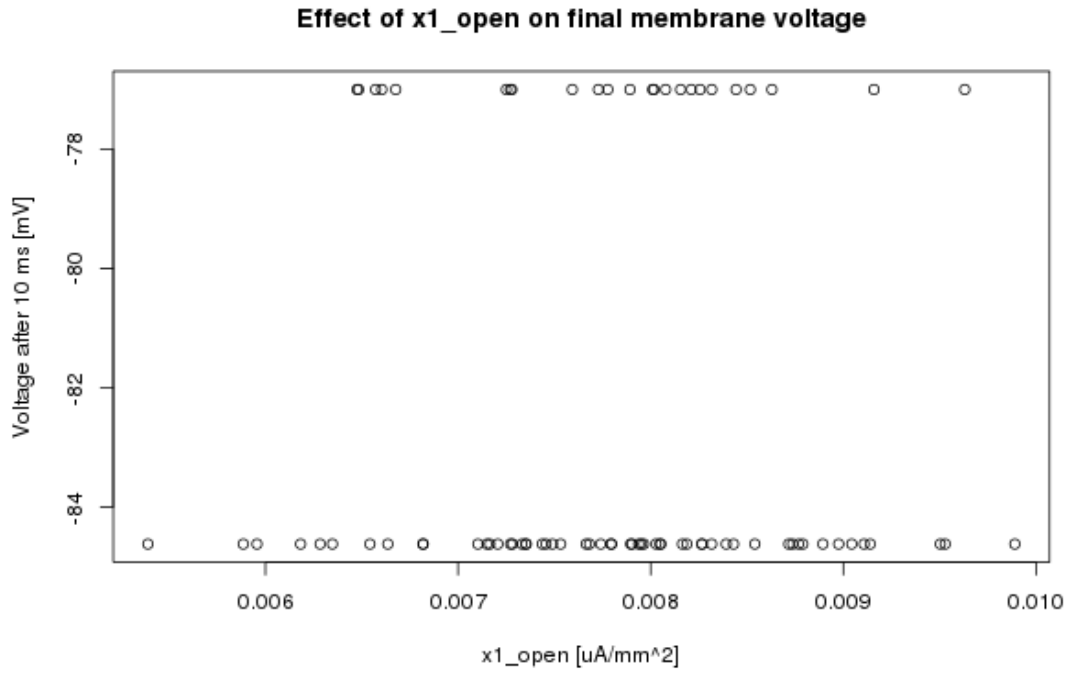

Figure 1: Results from the Simulation Experiment

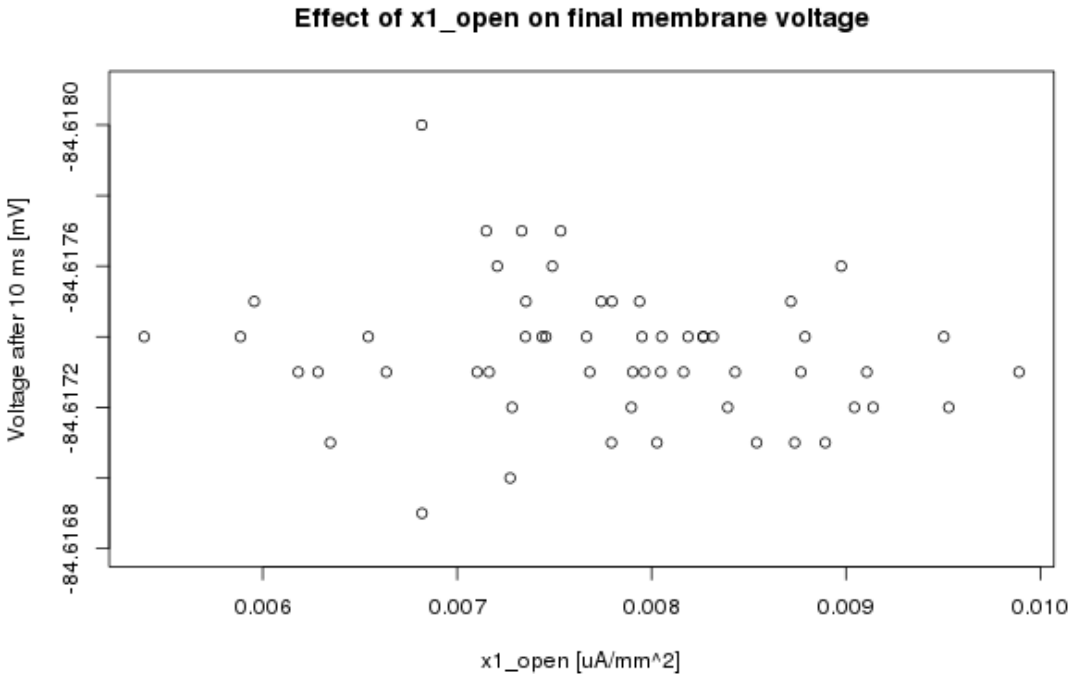

Figure 2: Results from the Simulation Experiment (with restricted y-axis range)

3.1.5 Physiome Model Repository support for rendering CellML models with uncertainty

The Physiome Model Repository (PMR) is a software package that allows modellers to share and version control mathematical models. The software is used to run the public repository at http://models.physiomeproject.org/. One feature of the Physiome Model Repository is that it can graphically display the mathematical relations in a CellML model. It is important that the Physiome Model Repository be capable of displaying uncertain parameters so modellers can easily see what uncertain parameters are included in the model. The Physiome Model Repository was extended so that uncertain parameters are displayed with the variable name on the left side, and the distribution on the right; distributions specified as *pdf*s are displayed as *pdf* (λx: expression), while realisations are shown as the vector of realisations.

An example of this rendering can be seen at http://models.cellml.org/e/103/beeler_reuter_1977.cellml/@@cellml_math in the component time_dependent_outward_current (see also Figure 3).

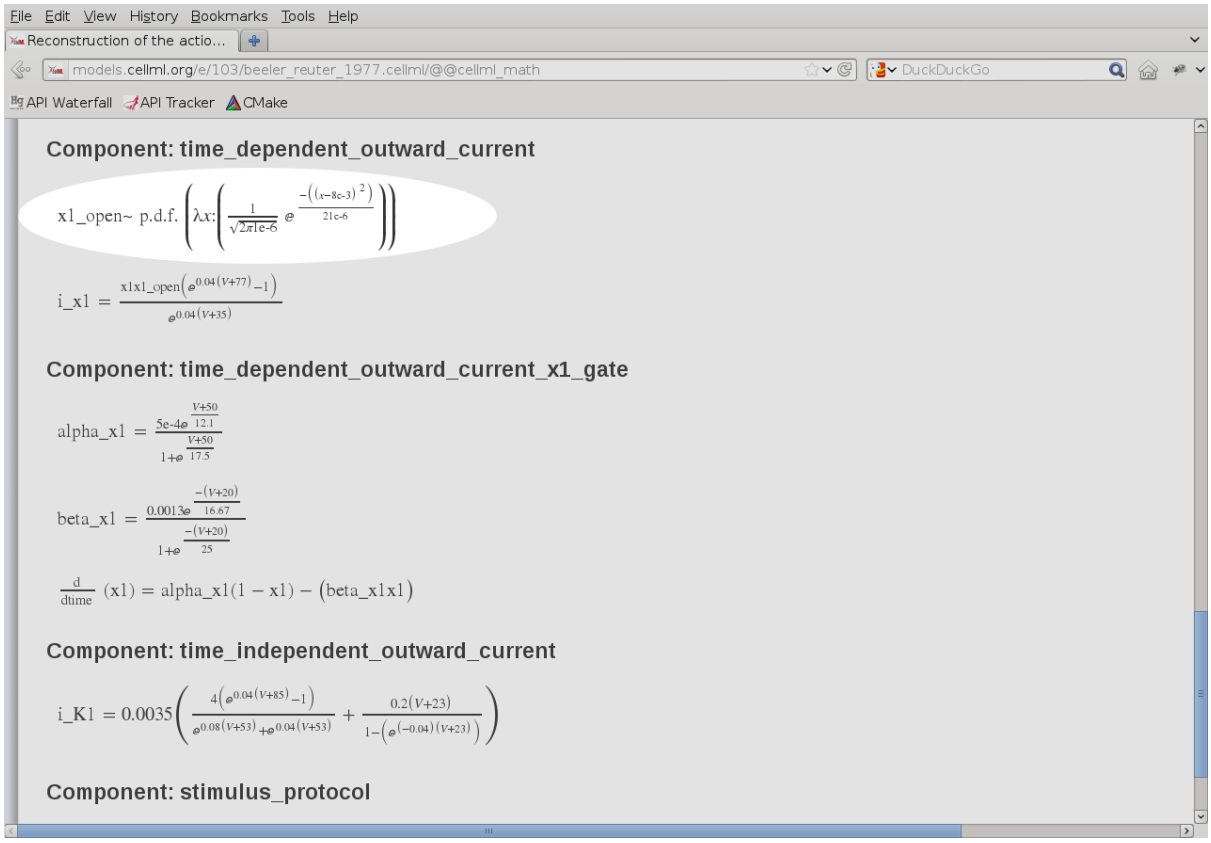

Figure 3: Screenshot showing PMR rendering of a probability density function. Image brightness / contrast has been modified to highlight the expression of interest.

3.2 A draft for CellML 1.2: Support parameter uncertainty more cleanly

A proposal has been put forward for a new version of CellML, CellML 1.2.[7] This proposal is structured as a precise but general purpose proposal that allows a wide range of mathematical expressions to be represented, with secondary specifications that describe specific sets of mathematical forms that can be supported by tools. One such secondary specification draft[8] was put together for models that describe systems of differential-algebraic equations and that may include parameter uncertainty and discontinuities at which variables normally governed by rate laws jump to a particular computed value.

One major advantage of this proposal over CellML 1.1 is that it makes use of Content MathML 3, which allows OpenMath Content Dictionaries to be referenced; Content Dictionaries define a series of mathematical symbols and explain their mathematical interpretation. The secondary specification proposal including uncertainty defines a new Content Dictionary called uncert1 containing the same uncertainty operators as described above for CellML 1.1.

The core specification proposal also adds a type attribute, which allows variables to have a specified type other than scalar. This will allow secondary specifications to be defined in the future for directly defining multivariate distributions using probability density functions involving vector expressions.

3.2.1 A prototype implementation of CellML 1.2 proposal with uncertainty

The draft CellML 1.2 proposal and secondary specification for models with uncertainty were implemented in a software package, available from https://github.com/A1kmm/cellml-testbed.

This package includes a processor, which converts models in the proposed XML-based format into the internal data structures used by the tool. As part of this processing, Content MathML 3 is converted into the strict form, allowing for simplified additional processing.

In addition, the package allows numerical simulations to be performed on the model. As in Section 3.3.1, uncertain parameters are sampled numerically as the simulation is run.

3.3 Supporting parameter uncertainty in FieldML 0.5

FieldML 0.5 is a specification for describing fields (where a field is a function that varies over an arbitrary domain such as space and / or time, with a value that could, for example, be a scalar, vector, matrix, or tensor). It is described in an academic paper which is currently under review, but which is available in the interim from the authors of this report on request.

A proposal for supporting parameter uncertainty in FieldML 0.5 has been put forward. Because FieldML 0.5 does not have facilities for describing arbitrary mathematical

[7] http://www.cellml.org/Members/miller/draft-normative-spec-andrews-preferred/toplevel.xhtml
[8] http://www.cellml.org/Members/miller/draft-secondary-spec-uncertainty/toplevel.xhtml

expressions, the proposal has focused on describing uncertainty by listing realisations, rather than by providing a mathematical description of the distribution.

FieldML 0.5 supports the concept of evaluator pipelines. An evaluator pipeline works like a function that takes arguments (which are formally specified using a construct called an argument evaluator), and produces a result through the composition of individual evaluators. The available evaluators are:

1. External evaluators, which carry out externally defined computations (for example, to perform an interpolation).
2. Parameter evaluators, which look up result values in a table of data using their arguments as indices into the table. This table of data might be a densely or sparsely packed array of data in HDF5, or a list of numbers appearing inline in the FieldML file, or in an external text file. Since some commonly used data formats such as VTK are usable as text files, this allows FieldML to wrap some other data formats.
3. Piecewise evaluators, which delegate the computation to a different evaluator pipeline depending on the value of an argument.
4. Aggregate evaluators, which produce an aggregate type by evaluating a value across a finite domain (called an ensemble in FieldML 0.5). For example, an ensemble might list all rows of a vector; an aggregate evaluator might then aggregate a scalar value over each row of the vector, producing a row vector.
5. Reference evaluators, which refer to other evaluators.

All evaluator types provide the option to optionally bind an evaluation pipeline to some or all arguments (effectively substituting those arguments for the specified evaluation pipeline); this is most commonly used with reference evaluators, where the arguments on the evaluator being referenced act like the function arguments, and the bindings to those arguments act like the specific parameters supplied for those arguments in a given call to the function.

Within this existing framework, support for uncertainty was added by adding in a new external evaluator called random.0d.equiprobable.tag. This external evaluator is defined to have the same behaviour as the identity function, except that when applied to an ensemble argument, it denotes that argument as being a randomly sampled value. This randomly sampled ensemble value may be used as a 'realisation identifier' to select which of a finite number of realisations has been sampled; a parameter evaluator may be used to map from this realisation identifier to the values of the realisation. By using an aggregate evaluator over an additional argument to the parameter evaluator, multivariate distributions may be described.

3.3.1 Solving FieldML 0.5 models with parameter uncertainty

An existing tool, the FieldML Scala testbed, was extended to support the approach to parameter uncertainty described in Section 3.3. This tool is available from https://github.com/FieldML/FieldML-testbed-scala/.

The extension allows for an arbitrary number of argument evaluators to be tagged using random.0d.equiprobable.tag, and causes each of these argument evaluators, if they are left

unbound when the model is evaluated, to be sampled randomly yielding a value from their corresponding ensemble type.

3.3.2 Creating a FieldML 0.5 model of an uncertain field of fixed geometry

To demonstrate that FieldML 0.5 can be used to describe multivariate distributions, and that those multivariate distributions can be used to supply the parameters to a mesh of continuous interpolators, a simple mesh model was defined and made available at http://models.fieldml.org/w/miller/uncertaingrid (Figure 4). This model was then visualised using the FieldML Scala testbed, using fixed positions for the mesh nodes, with the scalar field value visualised using heatmap colours (Figure 5). Nine different realisations of the model were sampled, each giving a different pattern on the heatmap. Each sample contains correlated values for the parameter at 25 different nodes, and so the nodal values are not all independent of each other.

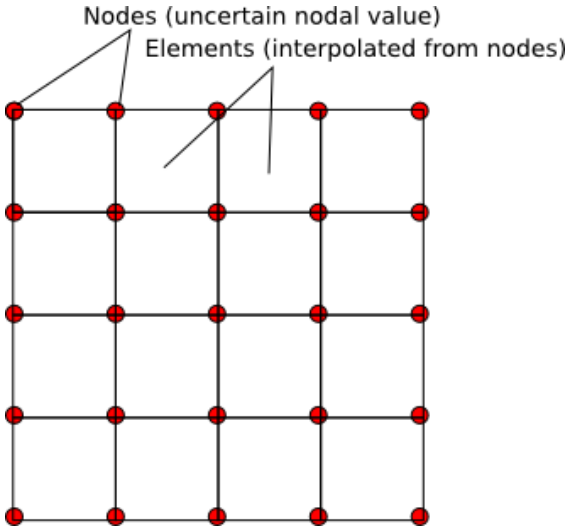

Figure 4: The arrangement of the nodes and elements in the fixed geometry example

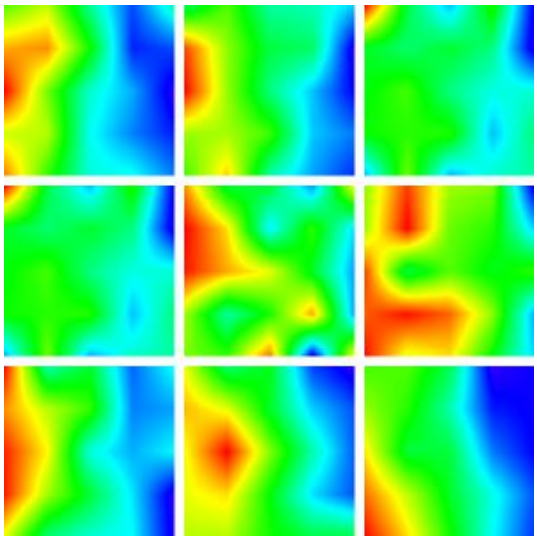

Figure 5: Nine realisations of the field, visualised as a heatmap

3.3.3 Creating a FieldML 0.5 model of an uncertain geometry

Another useful application is to be able to describe an uncertain geometry by providing different parameters to an interpolation that defines the nodal position vectors from the chart coordinate and the element. Using data supplied by Pablo Lamata (Department of Computer Science, Oxford University), five different realisations of heart shape were encoded into HDF5 and referenced from a FieldML 0.5 files defining how the data in the HDF5 file is used to interpolate over the mesh (Figure 6).

This model is, at the date of this report, available to registered users of the Anatomical Models Database[9], a database created as part of the FP7 funded EuHeart project, but will, once restrictions on the further dissemination of this data are lifted, also be made publicly available in the Physiome Model Repository.

Geometric field values can be computed using the FieldML Scala testbed, given the element identifier and chart coordinates.

In addition, the FieldML Scala testbed can be used to resample, from the tricubic Hermite interpolation field, a higher resolution trilinear Lagrange interpolated mesh represented in FieldML. This resampled FieldML file can be loaded directly into the cmgui tool for visualisation.

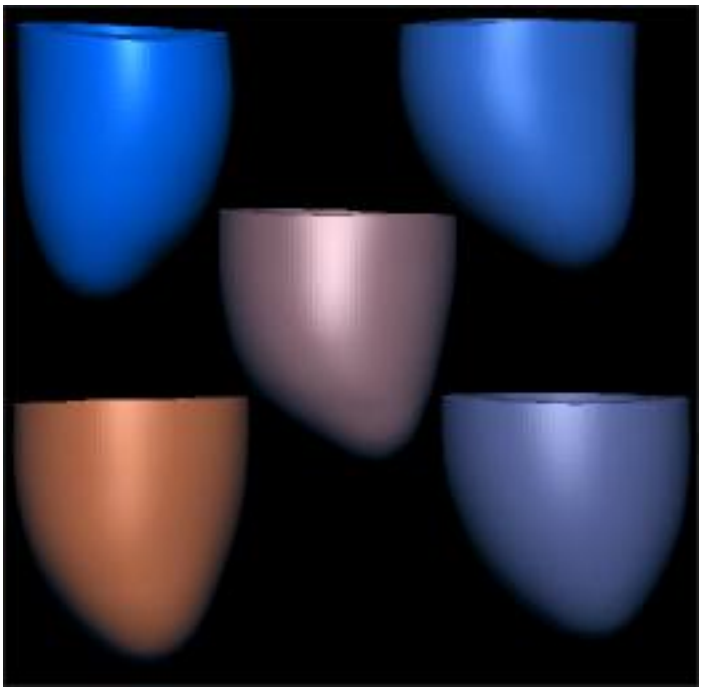

Figure 6: The five realisations from the Lamata cardiac model, resampled on a finer mesh and displayed using cmgui

9 http://amdb.isd.kcl.ac.uk:8080/

3.4 Extending FieldML for better parameter uncertainty support

Work is underway to develop a specification for a future version of FieldML. This specification is aimed at addressing a number of shortcomings with FieldML 0.5, to ensure that wider range of VPH-Share relevant models can be represented.

Prototype software for a draft version of this specification, which allows models to be parsed into an internal data format, is available at https://github.com/A1kmm/declarative-fieldml-prototype. This draft version makes major, non-incremental changes to FieldML designed to satisfy as many use cases as possible. Features supported by the draft language include:

1. Giving every field explicitly defined arguments, which are applied to provide the values for the arguments. Using a more traditional function application paradigm in the language makes it easier to define and use a library of mathematical operators, which will allow uncertain parameter values to be defined using probability density functions in the future.
2. Adopting a constraint programming approach (as in CellML and Prolog). Under this modelling paradigm, models are expressed as the composition of a series of constraints on variables; in the case of FieldML, variables may be fields over discrete or continuous domains. This approach makes it more natural to specify parameter uncertainty, because constraints can specify a 'distributed as' relationship between a variable and a distribution, as in the CellML extension, and will make it possible to specify Partial Differential Equation (PDE) models with parameter uncertainty in FieldML.
3. Introducing a powerful, state-of-the-art static type system for specifying the domains over which fields are defined, inspired by the Haskell type system, using Hindley-Milner type inference to allow type constraints to be placed and statically checked but without requiring every single sub-expression to be given a type. Type parameters allow generic, reusable model components to be written and reused (for example, across multiple organ systems). Type classes and instances allow types to be placed in classes, with all members of a class providing certain values / fields, so fields can be written that are reusable across all members of a type class. This will allow more powerful models containing uncertainty to be written, and for numerical solvers to more efficiently process models, including those containing descriptions of parameter uncertainty.
4. Providing a range of ways to define domains in terms of other domains. Real (with arbitrary physical units) is a built in domain used as a starting point. Domains can be cloned (making a new domain, for example, one dimensional space or time can be constructed from the real domain), put into a Cartesian product (for example, to combine three one dimensional space domains into three dimensional space), connected (for example, to specify that the top-right corner of one mesh element is the same point as the top-left corner of an adjacent element), subsetted (for example, to make a unit line domain containing points between 0 and 1 from the real domain, or to make a triangle from a unit square domain), put into a disjoint union (allowing the definition of domains where points are from one of several non-overlapping sets), or built into a field signature domain defining the set of all functions between two

domains. This powerful system for defining domains allows fields containing uncertainty to be defined over a wide range of domains. A demonstration of representing uncertainty using this approach is available in an earlier more modest prototype, available at https://github.com/codecurve/FieldML-Haskell-01.

5. Providing a more powerful namespace and import system which allows models containing uncertainty to be organised so that details specific to one part of the model are encapsulated in a namespace, but can, if that part needs to be extended, be imported into another part of the model.

4 ESTIMATES OF EFFORTS REMAINING

There are a number of aspects of task 5.3 which need to be completed to maximise the benefits to be captured from the task.

Current estimates are that these activities will be completed by the scheduled date, Project Month 30 (i.e. in August 2013), with overall effort expended on this task falling within the total effort allocation of 30 person-months.

The remaining aspects to be completed are:

1. To improve the declarative FieldML prototype so it can verify that identifiers used in models are valid, perform Hindley-Milner type inference and check that the domain annotations on the model are correct, and transform the model into a form in which it can easily be used by solver software.

2. To encode, in FieldML 0.5, a model of an uncertain strain field over a human femur, demonstrating the utility of FieldML for describing uncertainty in orthopaedic applications.

5 REFERENCES

1. Miller A, Britten R, Nielsen P. Declarative representation of uncertainty in mathematical models. PLoS One. 2012; 7(7).