D2.1: Report on multiscale UQ algorithms based on non-intrusive MC and semi-intrusive MC and mapping thereof in UQPs

Due Date	14 June 2019
Delivery	14 June 2019
Submission of updated version	N/A
Lead Partner	UvA
Dissemination Level	Public
Status	Final
Approved	Yes
Version	V2.0

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 800925.

DOCUMENT INFO

Date and version number	Author	Comments
17.04.2019 v0.1	A. Hoekstra	Skeleton, as sent to writing team
18.04.2019 v0.2	A. Nikishova	Addition of main text
20.04.2019 v0.2	D. Ye	Addition of main text
25.04.2019 v0.2	W. Edeling and D. Commelin	Addition of main text
28.04.2019 v0.2	D. Wright and R. Richardson	Review and providing input and text to next authors
30.04.2019 v0.2	D. Groen and A. Hoekstra	Addition of main text
10.05.2019	A. Hoekstra	Major edits with input from all other authors
22.05.2019	A. Hoekstra	Final edits
03.06.2019	A. Hoekstra	Update after internal review

CONTRIBUTORS

- A. Hoekstra (UvA) – Editor, author
- A. Nikishova (UvA) – Author
- Dongwei Ye (UvA) – Author
- Wouter Edeling (CWI) – Author
- Daan Crommelin (CWI) – Author
- D. Groen (UBRU)
- D. Wright (UCL)
- R. Richardson (UCL)

TABLE OF CONTENTS

1 Executive summary 4
2 Algorithms and Uncertainty Quantification 4
2.1 Introduction 4
2.2 Algorithms for Multiscale Uncertainty Quantification 5
2.2.1 Introduction 5
2.2.2 Non-intrusive methods 6
2.2.3 Semi-intrusive Multiscale methods 7
2.3 Uncertainty Quantification Patterns 9
2.3.1 From MMSF to UQPs 9
2.3.2 Toward UQPs 10
2.3.3 UPQ2: Semi-Intrusive acyclic 12
2.3.4 UPQ3: Semi-Intrusive cyclic 14
2.3.5 Bringing everything together and towards more advanced UQPs 17
2.4 From UQPs to the VECMA toolkit 19
2.4.1 Decomposing UQPs into elements 19
2.4.2 Implementing UQPs 20
2.4.3 Key considerations for implementation and execution 21
3 Conclusions and future work 22
4 Annexes 23
4.1 The Multiscale Modelling and Simulation Framework 23
4.2 Examples of applying semi-intrusive multiscale UQ algorithms 26
4.2.1 1D reaction-diffusion system 26
4.2.2 Gray-Scott model 28
4.2.3 ISR2D 29
5 References 31

1 Executive summary

During year 1 of the project WP2 has focussed on further developing the concept of Uncertainty Quantification Patterns (UQPs). Firstly, a family of semi-intrusive multiscale UQ algorithms is proposed, and partly tested on a set of cyclic multiscale models, demonstrating that indeed such semi-intrusive methods are capable of estimating the output uncertainties in a reliable way, while at the same time reducing the required execution time of the overall multiscale UQ.

Based on this experience, and guided by the Multiscale Modelling and Simulation Framework, a family of UQPs is designed. The main result is a distinction in control structure, leading to five different variants. All these variants can then be further refined using optimisation strategies.

An interesting and important result is that all variants of UQPs build upon the most basic 'black box' UQP1. This UQP has been implemented in the VECMA toolkit in WP3, relying on generic UQ Elements (UQEs) that can be realised in software. The design considerations for the software developments, based on the UQPs and the concept of UQEs, are also reported in this deliverable.

WP2 is on track, next steps will be to further explore and develop the UQPs and to realise multiscale V&V.

2 Algorithms and Uncertainty Quantification

2.1 Introduction

The objectives of Workpackage 2, *Algorithms & Formalisms*, is to develop algorithms for Multiscale Uncertainty Quantification (UQ), to generalize these algorithms to Uncertainty Quantification Patterns (UQPs), to scale and optimize them for existing petascale and emerging exascale architectures, and to develop methods for verification and validation of multiscale simulations and capture these into Verification and Validation primitives (VVPs).

Task 2.1 (Multiscale UQ algorithms based on non-intrusive MC and semi-intrusive MC) and task 2.2 (UQP development based on task 2.1) started as planned, and task 2.1 was finalised as planned in M12 of the project. The achievements of these tasks are presented in sections 2.2 - 2.4.

2.2 Algorithms for Multiscale Uncertainty Quantification

2.2.1 Introduction

Computer modelling is widely used in science and engineering to study systems of interest and to predict their behaviour. These systems are usually multiscale in nature, as their accuracy and reliability depend on the correct representation of processes taking place on several length and time scales [1–4]. Moreover, these multiscale systems can be stochastic, since there are always some unresolved scales, whose effects are not taken into account due to lack of knowledge or limitations of computational power [2,5]. Additionally, measurements of model parameters, model validation, or initial and boundary conditions rarely can be achieved with perfect accuracy [6]. Therefore, the model results inevitably contain uncertainties, and one should estimate their magnitudes by applying a forward uncertainty quantification (UQ) method.

Usually a distinction is made between intrusive UQ methods, where one substitutes the original model with its stochastic representation, and non-intrusive methods, where the original model is used as a black-box [7,8]. Intrusive methods are efficient and relatively easy to apply to linear models, e.g. [9]. This, however, represents only a relatively small class of models. They can be applied to non-linear models as well, but the solution of the resulting equations may become very demanding. Non-intrusive methods can be applied to any type of non-linear model. However, if a single model run requires large execution times, these UQ methods may be ineffective, or even computationally intractable.

We introduce a family of semi-intrusive UQ algorithms for multiscale models, see also [10]. These methods are called semi-intrusive, since they are intrusive only on the level of the multiscale model, that is, in the way the single scale components are coupled together. The single scale components themselves however are treated as black-boxes, see Figure 1.

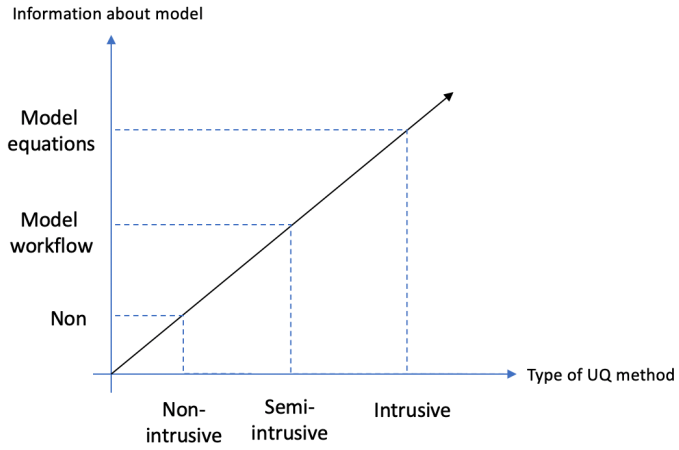

Figure 1: Intrusiveness of UQ methods.

In the semi-intrusive metamodeling approach a surrogate model substitutes the most expensive single scale model. The metamodel can be constructed by applying, for example, a data-driven approach, like Gaussian process regression [11–14], or using a spectral approach, for instance, based on polynomial chaos [15–17]. A comparison of polynomial chaos and Gaussian process surrogates is presented in [18]. Since only one component of the multiscale model is approximated by the surrogate, the resulting error in the model output can be small enough to still be able to obtain reliable uncertainty estimates. However, it is expected that this strongly depends on the sensitivity of the output of the multiscale model on that of the single scale component and the method used to build a surrogate of that single scale component.

In the semi-intrusive Monte Carlo (SIMC) method the number of samples for the computationally intensive part of the multiscale model (usually microscale dynamics) is reduced. This leads to a decrease in the computational time for the multiscale UQ. A cross validation, which is part of the method, controls the level of sub-sampling and hence the accuracy of the estimates of uncertainty.

Both the SIMC and metamodeling methods are based on an approximation of the original model output. The important difference is that in the SIMC we construct a local data-driven surrogate at each call of the micro model. This allows to estimate the effect of the approximation on the uncertainty estimates, and adjust the sample size needed. Moreover, in the case of a deterministic multiscale model, these local surrogate models have much lower input's dimensionality in comparison to global surrogates as used in the metamodeling methods, which equal to the dimensionality of uncertain inputs, since at each micro model run a single output can be obtained with a particular value of these uncertain inputs.

Before describing in more detail the semi-intrusive methods, a short account of non-intrusive methods (as implemented in WP3) is provided in section 2.2.2.

2.2.2 Non-intrusive methods

Non-intrusive uncertainty propagation methods consider the entire (multiscale) system as one black box, see Figure 2. The main advantage is that the (legacy) simulation code is left completely untouched, hence the name ‘non-intrusive’. This allows users to quickly add a UQ component to their existing simulation framework.

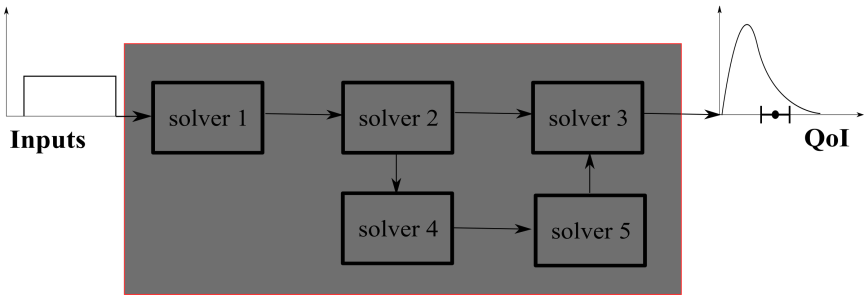

Figure 2: Schematic of non-intrusive uncertainty propagation through a multiscale system of coupled single-scale solvers, mapping input distributions to a distribution of any output Quantity of Interest (QoI). The propagation technique is agnostic with respect to the structure of the multiscale system, and treats it as a black box.

To facilitate this, several non-intrusive methods are considered in VECMA, specifically the Monte Carlo, Polynomial Chaos and the Stochastic Collocation methods [7]. All these methods follow a similar pattern, namely:

1. Specify the input distributions and draw samples (create a so-called design-of-experiment).
2. Run the ensemble.
3. Perform post-processing analysis.

This common structure is exploited by creating generic UQ Elements for each step. For instance, the core functionality required by EasyVVUQ, as developed and implemented in Workpackage 3, of step 1 (sampling), is contained in a `BaseSamplingElement`. The sampler of any given method, e.g. Stochastic Collocation, inherits all base functionality and only adds what is specific to that method. This setup makes it easy for an 'expert user' to quickly add other propagation methods.

2.2.3 Semi-intrusive Multiscale methods

The semi-intrusive methods for multiscale UQ are a family of algorithms, which employ the structure of the multiscale models in order to perform an efficient UQ, that is, estimating the uncertainties with comparable quality as the black box MC method, but with a substantially reduced execution time. According to the Multiscale Modelling and Simulation Framework [19–21], instead of considering the whole multiscale model as a black-box, the model can be seen as a collection of coupled single scale black-box systems. Thus, the semi-intrusiveness of the methods boils down to a limited inspection of the multiscale model, which is only up to the level of single scale components and their coupling. Below the main ideas behind the semi-intrusive UQ methods is described. First results on applying these methods on test cases and actual applications from the VECMA application portfolio are provided.

2.2.3.1 Semi-intrusive Monte Carlo

The semi-intrusive Monte Carlo (SIMC) is a Monte Carlo method with a reduced number of samples of the expensive component of the multiscale model, see Figure 3. The remaining samples are obtained by interpolation. Usually the interpolation method produces results which are not exact to the micro model response. Therefore, a statistical cross-validation is applied to test whether the interpolation does not lead to a large error in the estimates of uncertainty: the error is compared to the confidence interval of the N_μ MC estimate, and then, our algorithm accepts the SIMC results when the error is smaller than the confidence interval and the MC results, otherwise. All details can be found in [10].

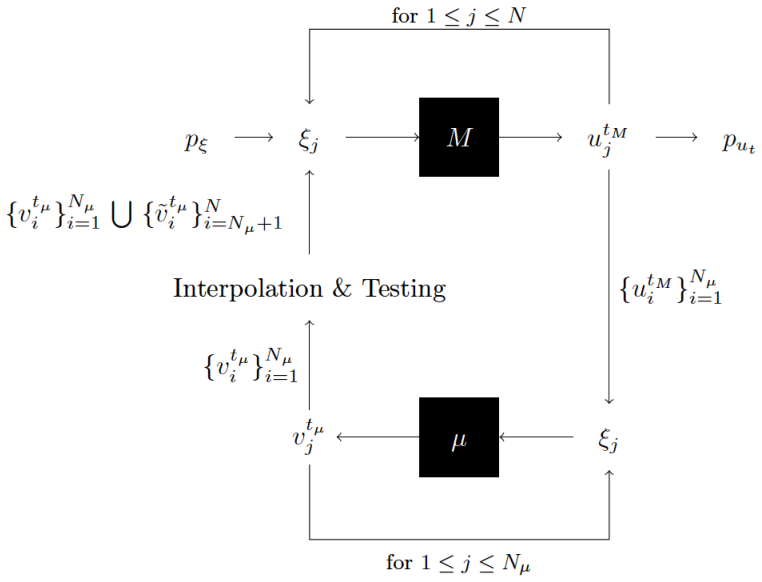

Figure 3 Semi-intrusive Monte Carlo method with a smaller number of samples of the expensive microscale model

2.2.3.2 Metamodeling of a single scale model

Surrogate modelling is a common approach to perform an efficient UQ for computationally intensive systems at a reduced amount of time [16,18]. The idea of these methods is to substitute the original system by its surrogate, which produces a similar output, but their computational time is lower. In the semi-intrusive multiscale metamodeling method, these techniques are applied to a single scale component, which takes the largest portion of the computational time [22]. In this way, the error introduced by the approximation is expected to be small when estimating the uncertainties of the multiscale model.

Figure 4 shows an example where the micro model is substituted by a surrogate. The rest of the multiscale model has the original form. However, since the micro model produces an approximate

result, the output of the macro model is not the same as the original model. In this method, the error will always depend on the details of the model. It depends on the properties of the micro model, for example, smoothness, which determines how difficult it will be to approximate the original single scale model. Additionally, the error in the estimates of uncertainty also depends on how sensitive the result of the macro model is to the output of the micro model which is replaced by a surrogate. If, for instance, this sensitivity is low, it is reasonable to expect that the error introduced by the approximation is small. Of course, the error also depends on the method with which the surrogate is build.

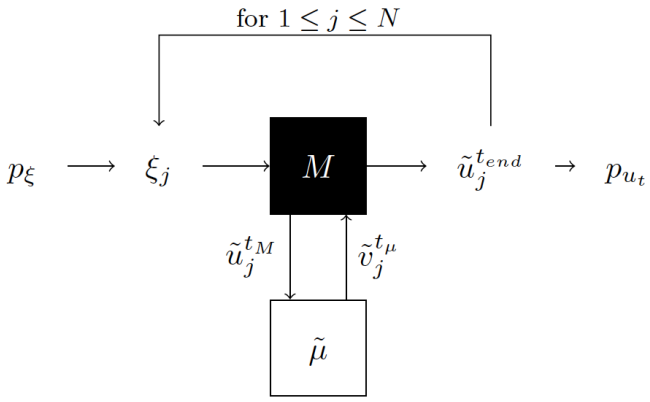

Figure 4: Semi-intrusive multiscale metamodeling uncertainty quantification

2.3 Uncertainty Quantification Patterns

2.3.1 From MMSF to UQPs

With VECMA, we will address the two main aspects of UQ: uncertainty propagation, and sensitivity analysis (SA). The former deals with how uncertainties in model inputs and parameters propagate to model output uncertainties, i.e. it is about quantifying output uncertainty caused by input uncertainty. SA tackles the question of which input uncertainties contribute most to the overall model output uncertainty, in the (common) situation of multiple uncertain inputs. The two are closely related, in particular, the high computational cost of performing uncertainty propagation with many inputs can be reduced by focusing on the inputs found to be most important by SA. Uncertainty Quantification Primitives (UQPs) are designed to support both UQ and SA in multiscale applications, and which are scalable to the exascale. They provide the basic building blocks, to create tailored UQ for multiscale models. In terms of implementation, we anticipate the UQPs to be provided as generic templates, which can then be customized and aggregated to create a dedicated UQ procedure for multiscale applications.

As discussed in section 2.2 it is possible to exploit the structure of a multiscale model to reduce the computational requirements of performing a multiscale UQ, the so-calling semi-intrusive UQ, while still being able to obtain good estimates of the uncertainties. An important assumption in VECMA is that computational requirement of a single multiscale simulation can be so high, that even on Tier0 machines performing a non-intrusive UQ on the multiscale model would be prohibitive, necessitating the need for algorithmic improvements. The Multiscale Modelling and Simulation Framework (MMSF) has been developed in earlier EU funded projects, and its main features are summarized in appendix 4.1.

2.3.2 Toward UQPs

Central to VECMA are multiscale models, consisting of several single-scale models and scale bridges coupled together. We will exploit this structure to boost the efficiency of UQ for multiscale models. The starting point is the availability of non-intrusive methods such as (quasi-)Monte Carlo (MC/qMC), Stochastic Collocation (SC), and Non-Intrusive Spectral Projection (NISP). For SA, suitable approaches are e.g. Sobol's method and Saltelli's very efficient algorithm to approximate the first order Sobol sensitivity indices, posterior density approximations and MCMC-based estimates. A first set of UQPs consists of applying these methods to the multiscale model as a whole. We distinguish between UQPs employing basic MC and those using more advanced methods such as qMC, SC and NISP. To increase efficiency, a further set of UQPs that employ existing non-intrusive methods at the level of single-scale models (or scale bridges). These UQPs combine single-scale computations in accordance with the multiscale structure to obtain UQ (or SA) results for the multiscale model. We refer to such UQPs as ‘semi-intrusive’.

The MMSF allows a characterisation of multiscale models and multiscale computing along several dimensions

- Interaction regions, i.e. scale overlap or scale separation, both temporal and spatial (see Figure 16), leading to 5 different options;
- Single domain versus multi-domain, leading two 2 options;
- Multiscale computing, acyclic versus cyclic (see Figure 18) and the coupling topologies (one, versus a fixed or dynamic number of instantiations and one, versus a fixed or dynamic number of synchronisation points, see Figure 19) leading to 9 options.

In principle this would lead to $5 \times 2 \times 9 = 90$ different types of multiscale simulations (when coupling only two single scale models together). This is relevant for details of the multiscale modelling, and for multiscale computing, but in designing UQPs we need not consider all these permutations.

As the main goal of the UQPs is to try to exploit the structure of the multiscale model to reduce the computational cost of a multiscale UQ, it seems logical to first consider the coupling topologies to define UQPs, and then invoke where needed the notion of interaction regions, and finally, of the computational domain. Consider a prototypical multiscale model consisting of two submodels, A and B, coupled together in the most general sense, see Figure 5. Both A and B take uncertain inputs (initial conditions, boundary conditions, parameters, the blue incoming arrows) and both produce Quantities of Interest (QoI) with uncertainties (the orange outgoing arrows).

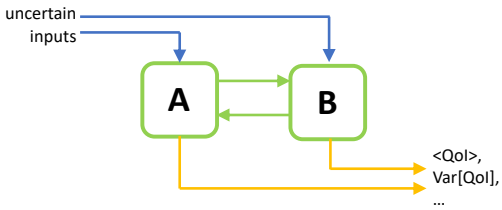

Figure 5: prototypical multiscale model.

The simplest UQP, called UQP1, is to not exploit the structure at all, and consider the multiscale model as a black box that has inputs and produces outputs, see Figure 6. In UQP1 the UQ and SA is performed by using non-intrusive methods on the application as a whole, and quantifying the uncertainty relative to a Quantity of Interest (QoI) that resides in the final application output. By definition this approach cannot identify the contribution to the UQ from individual submodels of a multiscale application. SA will only identify contributions of inputs and parameters that are provided as external input to the overall multiscale application. The current VECMA toolkit provides all required functionality to implement UQP1.

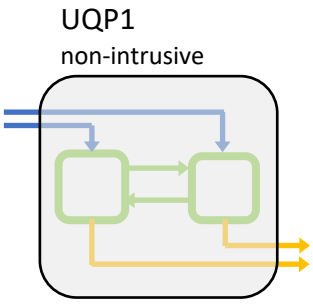

Figure 6: The non-intrusive UQP1.

The first main distinction that can then be made is acyclic versus cyclic multiscale models, leading to three main distinct classes of UQPs, see Figure 7. We call these control structure UQPs, where UQP2 relates to acyclic multiscale models and UQP3 to cyclic multiscale models. Both UQPs will implement semi-intrusive UQ approaches, meaning that the multi scale model is now opened up to

realise more efficient multiscale UQ algorithms, but the single scale models are still treated as black boxes.

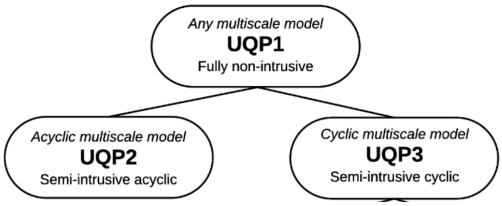

Figure 7: Three main classes of UQPs.

2.3.3 UPQ2: Semi-Intrusive acyclic

In case of acyclic structure, uncertainty propagates in one direction through the multiscale model. Output uncertainty of one single component creates input uncertainty of another component. UQP2 performs non-intrusive UQ and SA on consecutive single components. It enables investigation into how uncertainty propagates and becomes amplified within each component of the model.

The prototypical acyclic multiscale model is shown in Figure 8, where the main difference with Figure 5 is that now submodel A feeds into submodel B, but not vice versa. UQP2 can be realised as a serial application of UQP1, first to submodel A and then to submodel B, see Figure 9.

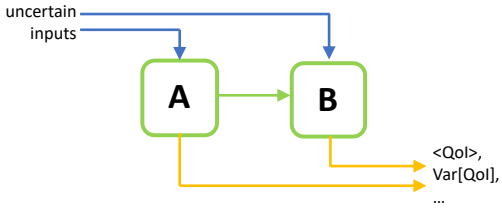

Figure 8: Prototypical acyclic multiscale model.

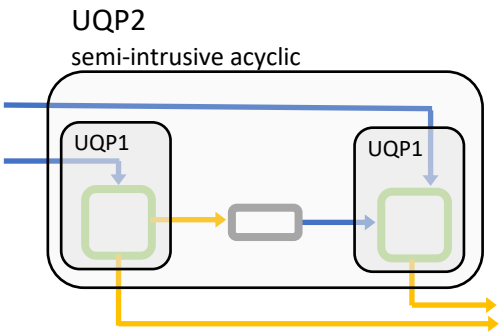

Figure 9: UQP2, semi-intrusive acyclic, as a composition of UQP1s.

Note a few important issues. First, after applying UQP1 to submodel A, the data to be sent to submodel B has now turned into an uncertain output (denoted by the orange line), which is then converted into uncertain input for submodel B (in the grey-lined box, resulting in the blue arrow into submodel B). Depending on details of submodels A and B, this operation can be far from trivial, e.g. capturing correlation between outputs of uncertain outputs of A, that need to be accounted for when creating the uncertain input for B. First results in a fusion application in WP4 demonstrate what such operation could mean.

Suppose that submodels A and B have n_A and n_B uncertain input parameters, and that the execution time for submodels A and B is T_A and T_B respectively. Moreover, suppose that for each uncertain parameter we must sample σ independent samples from the distribution of the uncertain parameters, when performing a 'black box' QMC UQ. If we would do this for the whole A → B multiscale model, this would result in an execution time

$$T_{UQP1} = \sigma^{(n_A+n_B)}(T_A + T_B), \quad [1]$$

basically meaning that the black box UQ requires $\sigma^{(n_A+n_B)}$ runs of the multiscale model. If we would now apply UQP2, so first do the UQ on A, and then propagate the uncertain output to B and do a UQ on B, we find that, assuming that now for B we have $n_B + n_{A\to B}$ uncertain parameters (namely also the information sent from A to B)

$$T_{UQP2} = \sigma^{n_A}T_A + \sigma^{n_B+n_{A\to B}}T_B, \quad [2]$$

where we ignored the time required to convert uncertain output from A to input for B. The speedup that can be achieved by UQP2 then becomes

$$\frac{T_{UQP1}}{T_{UQP2}} = \frac{\sigma^{(n_A+n_B)}(T_A+T_B)}{\sigma^{n_A}T_A+\sigma^{n_B+n_{A\to B}}T_B}. \quad [3]$$

If n_A and n_B are comparable, and in the limit of $T_A \gg T_B$ we find a speedup of σ^{n_B}, which can be substantial. And vice-versa, if $T_A \ll T_B$ the speedup becomes $\sigma^{n_A-n_{A\to B}}$. Depending on n_A and $n_{A\to B}$ the speedup can again be substantial, but in this limit UQP2 can also result in a slowdown if $n_{A\to B} > n_A$.

If T_A and T_B are comparable, say $T_A = T_B$, we find $\frac{T_{UQP1}}{T_{UQP2}} = \frac{2\sigma^{(n_A+n_B)}}{\sigma^{n_A}+\sigma^{n_B}+n_{A\to B}}$, and further simplifying by assuming that $n_A = n_B = n$ gives $\frac{T_{UQP1}}{T_{UQP2}} = \frac{2\sigma^{n}}{1+\sigma^{n_{A\to B}}}$, again resulting in potentially large speedups. For large enough σ we find $\frac{T_{UQP1}}{T_{UQP2}} = 2\sigma^{n-n_{A\to B}}$. In this limit, UQP2 will result in speedups as long as $n_{A\to B} \leq n$.

Referring to Figure 19, the coupling topologies, UQP2 will still be applicable if submodel A or B would have multiple fixed or dynamic instances. However, this would require a more extended 'wiring up' in software. It remains to be seen, in close collaboration with WP3 and WP4, if such applications are encountered, and what that would mean for expressing the UQP in software. Also note that submodels A and B could, in principle, be quite complex multiscale models by themselves. Therefore, if a multiscale model has an acyclic coupling somewhere, it can be decomposed into two large components and UQP2 could in principle be applied.

2.3.4 UPQ3: Semi-Intrusive cyclic

For cyclic multiscale models one could first explore if constructing a task graph as in Figure 22 and then applying UQP2 would be an option. Certainly, for a fixed number of synchronization points this is theoretically possible, but even if the number of cycles is dynamic executing the cyclic multiscale model as if it were a fully serialized acyclic model would be an option. In its full generality such cyclic model would then unfold as in Figure 10. An important additional feature now is that in cyclic models, the submodels A and B typically iterate over time (see the SEL, Figure 17) and must be considered stateful, meaning that in say iteration *i* of the cycle, process A[*i*] would update its state based on its state in iteration *i*-1 and input from B[*i*-1]. So, in principle, UPQ2 (Figure 9) could be applied, but the details of the data sent from A to B, including the states from one iteration to the next, that should now become uncertain outputs and uncertain inputs, will dictate to what extend this approach is useful. In VECMA we may explore this option for some applications, to understand the efficacy of this approach.

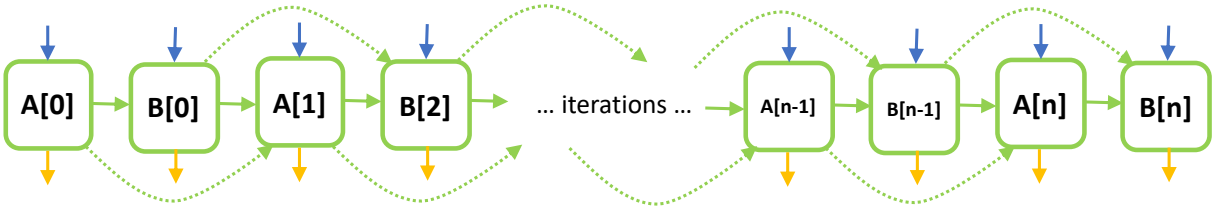

Figure 10: Fully serialised cyclic multiscale model. The dotted green arrows denote that the iterations of A and B are statefull, and that data is sent from A[i] to A[i+1] and B[i] and B[i+1]. The blue and orange arrows denote again the uncertain inputs and outputs.

In order to proceed we need to invoke a few more distinctions and assumptions. The next distinguishing feature of a multiscale model is in the interaction regions (see Figure 16). For the control structure and how in detail a multiscale simulation is executed, the main feature is if the timescales of submodel A and B overlap or are separated. In case the timescales are separated (so, interaction regions 1, 3.1 and 3.2), and assuming that submodel B is the slow process, this basically means that after every iteration of submodel B, submodel A is started and run to completion (this is the example shown in Figure 17). When the timescales overlap (interactions 0 and 2) the SELs of submodel A and B are coupled in the inner loop (so, $O_{i,A}$ to $(S,B)_B$ and $O_{i,B}$ to $(S,B)_A$). These considerations lead to a further distinction in UQP3, see Figure 11. Here we follow the naming as suggested in the VECMA proposal. However, this naming is part of an ongoing discussion during the final months of task 2.2 (UQP development) and follow-up task 2.4 (advanced UQP development).

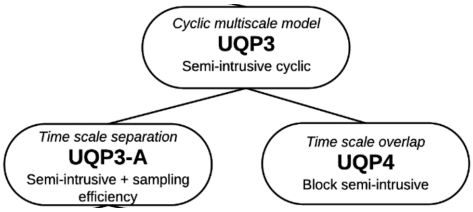

Figure 11: Further distinction in UQP3 between time scale separation (UQP3-A) and time scale overlap (UQP4)

For now, UQP4 seems to not open up to more advanced algorithms, and most probably the best candidate for now seems to be to try to apply the approach of UQP2, as suggested above, so applying UQP2 to the serialised workflow as in Figure 10. As written before, we will further explore this, in the realm of tasks 2.4 and driven forward by demands from applications in WP4 that would fall in this category.

For UQP3-A, so for cyclic multiscale models with time scale separation, we can continue to make additional assumptions, based on our experience in handling many multiscale applications that fall into this category. Usually, the Quantities of Interest in such multiscale models is derived from the slow dynamics. Assuming that submodel B implements the slow dynamics, the prototypical multiscale model for UQP3-A is as in Figure 12.

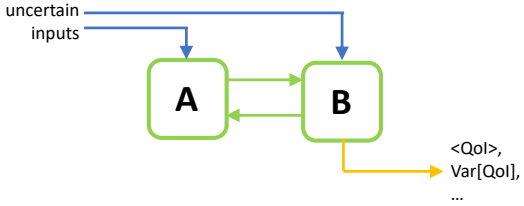

Figure 12: Prototypical cyclic multiscale model with time scale separation, where B is the slow process.

At the same time, the fast dynamics typically involves microscopic processes with huge degrees of freedom (e.g. explicit molecular dynamics simulations). Executing such microscopic model is normally already an expensive operation, and when coupling it to a slow process with which it is in quasi-equilibrium means that the fast process is executed to completion for every iteration of the slow process. This means that most of the execution time of such a multiscale model is spent on the fast processes.

In a UQ scenario for such a multiscale model, our working hypothesis is that in order to estimate the uncertainties in the QoI, some errors in the fast process simulations can be tolerated. If so, we could try to reduce the number of runs of the fast dynamics (subsampling) and estimate the remaining runs by interpolation. This is the approach followed in Figure 3. Another option would be to replace the fast dynamics altogether by a surrogate, see Figure 4. In the examples reported in section **Error! Reference source not found.** we have demonstrated that our working hypothesis is valid, and that as a result the execution time required for multiscale UQ for cyclic time scale separated problems can be drastically reduced.

First consider the case of subsampling and interpolation (as in Figure 3). The resulting UQP3-A is drawn in Figure 13. Interestingly, UQP3-A means adding an additional box between the fast and the slow dynamics, to orchestrate the subsampling, interpolation, and statistical testing of the interpolation, and then basically running a UQP1 on the resulting enhanced multiscale model. This additional box must be provided in some dedicated way, which depends on how the actual multiscale simulation is implemented. However, the basic functionality required for this subsampling box, should be made available via the VECMA toolkit.

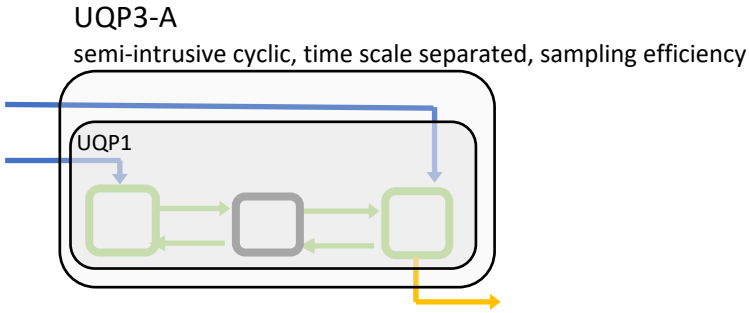

Figure 13: UQP3-A, semi-intrusive cyclic, time scale separated, sampling efficiency, as a UQP1 on a slightly changed multiscale model.

We can also consider replacing the fast dynamics altogether by a surrogate. In the proposal this was coined UQP3-B (and again, for now we maintain this terminology, before ending task 2.2 we intend to

revisit this). The resulting UQP3-B is drawn in Figure 14. Again, UQP1 is applied to a slightly changed multiscale model, where the fast dynamics is replaced by a surrogate model. Most software that is used to construct cyclic multiscale simulations (such as e.g. MUSCLE2) have a plug and play feature that would allow such a change. Than the main challenge is to construct the metamodel, based on the uncertain input parameters of the fast and the slow dynamics, and based on prior knowledge on where and how the slow dynamics would 'steer' the fast dynamics through its parameter space. Details of this are of course highly application specific, but having some generic software for popular classes of metamodeling approaches (e.g. building Gaussian Process based surrogates) in the VECMA toolkit, and being able to easily set up the workflow as implicated by UQP3-B is desirable for efficient multiscale UQ.

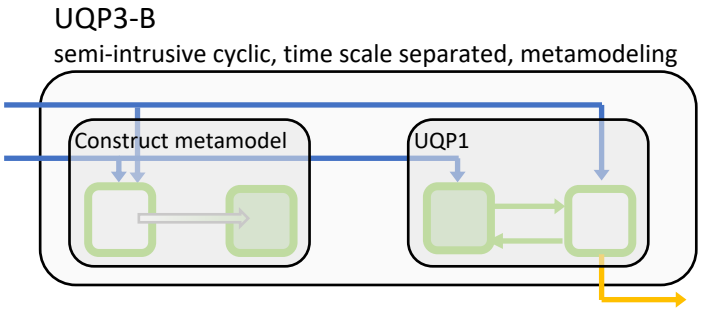

Figure 14: UQP3-A, semi-intrusive cyclic, time scale separated, with metamodelling, as a workflow where first the metamodel is constructed for the fast dynamics, followed by a UQP1 on a slightly changed multiscale model, where the fast dynamics is replaced by the metamodel.

2.3.5 Bringing everything together and towards more advanced UQPs

The discussions above leading to UQP1, UQP2, UQP3-A,B and UQP4 are in line with those that were originally described in the proposal. However, we have now substantiated them, clearly embedded them in the MMSF, and have shown how UQP2 – UQP4 could be composed out of UQP1, with additional functionality that should be embedded into the execution of the multiscale model while performing the multiscale UQ. For UQP2 we have also provided some estimates on performance, which need to be tested in practice and extended to the other UQPs. And for UQP3-A and UQP3-B we have shown some examples of applications using the ideas behind these UQPs to corroborate the working hypothesis that underlies the UQP3-A,B family (QoI on the slow dynamics, most computational effort in the fast dynamics, and tolerance in estimation of uncertainties).

To summarize, the UQPs introduced above have been a combination of what we in the proposal coined *control structure* UQPs and *optimization* UQPs. The control structure UQPs are based on the distinction between acyclic and cyclic multiscale models, and for the latter, in time scale overlap or

time scale separation. For acyclic multiscale models the interaction regions and the relevant coupling topologies (single, fixed, or dynamic number of instantiations of the submodels) do not seem to result in other multiscale UQ approaches (albeit in much more complex workflows), and UQP2 seems well defined (but see the discussion on optimisation UQPs below). For UQP3 and UQP4 more work is required (in task 2.2 and 2.4) to further explore the implications of the control structures.

The optimization UQPs that were discussed are in relation to subsampling of expensive fast dynamics and replacing a submodel by its surrogate. Another optimization could be to perform an intrusive UQ on a submodel, and replace that submodel by the intrusive UQ variant. Table 1 provides an overview of all possible UQPs along these two dimensions. The naming conventions are not yet agreed upon, the deliverable reflects the current thinking in the consortium. The UQPs in green have been discussed above and partly tested in VECMA (WP2, WP3, WP4). Those in orange are under investigation, and depending on the needs of the applications, will be further developed. Note that the optimisation dimension also allows combinations, e.g. combining subsampling (A) with metamodeling (B) is of course an option. If that would lead to further reductions of execution time of the multiscale UQ using subsampling or metamodeling alone remains to be seen.

Table 1: Overview of UQPs, along the control structure and optimization dimensions. The UQPs in green have been discussed in the main text, those in orange require further analysis and development. The red crosses indicate void combinations, and the question marks need further investigation.

		Control Structure				
		none	acyclic	cyclic		
				time scale overlap	time scale separation	
					single domain	multi domain
Optimization	None	UQP1	UQP2	UQP4	UQP5 (?)	UQP3
	sub sampling (A)	✗	?	?	UQP5-A	UQP3-A
	metamodelling (B)	✗	UQP2-B	UQP4-B	UQP5-B	UQP3-B
	instrusive single scale (C)	✗	UQP2-C	?	UQP5-C	UQP3-C

Finally, in Table 1 we have not made deeper distinctions in terms of the coupling topologies. So, we kept the cyclic with fixed or dynamic number of cycles together. Currently we don't see the need to split even further along this dimension. Likewise, for the number of instances of the single scale codes. As discussed above, in the acyclic case this can in principle be captured by UQP2. For the cyclic case this is a bit subtler. A cyclic time scale separated model with a single domain usually lies in interaction

region 3.1 (meaning that also the spatial scales are separated) and that a large number of instantiations of the microscale model is required, basically leading to the Heterogeneous Multiscale Model. So, UQP-5 already implies multiple instances of the microscale submodel. Moreover, from the ComPat project we know that efficiently executing such Heterogeneous Multiscale Model requires a surrogate model anyway, to prevent executing too many microscale models [31]. This suggests that the UQP5 family basically implies UQP5-B, and that the other UQP5 variants would be void. This requires deeper analysis. The UQP3 family was constructed with a few assumptions (mainly that the QoI are on the slow dynamics). This needs to be further explored and tested against our current applications portfolio, and against all other multiscale models that we are aware of.

2.4 From UQPs to the VECMA toolkit

In this section we describe how we map the UQPs proposed in WP2 to a working and applicable implementation for use by the VECMA project, as well as external users. This description reflects an ongoing learning process, which is grounded in fast-track UQ activities performed up to this point and will continue as the project progresses. We first summarize the low-level building blocks that we identify from the implementation perspective, so-called UQ elements (or UQEs), then describe how UQPs as a whole are reflected in the implementation. Lastly, we reflect on several key implementation and execution aspects of using the proposed UQPs.

2.4.1 Decomposing UQPs into elements

UQPs are algorithmic specifications of UQ that are generic, extensible and widely applicable. The algorithmic specification level is well suited for most scientific reflections, where the right conceptual granularity is key in retaining both a simple means of discourse, and enough detail to identify key differences. From the perspective of toolkit design and implementation, we work with a hierarchy of perspectives (e.g., from generic to application-specific) and levels (e.g., from full application to component to single lines of code). A fine-grained conceptual granularity is needed here, because fine-grained concepts lead to simple functional implementations, that are more likely to be reused in other places in the toolkit (i.e. help reduce duplicate code). To efficiently implement UQPs, we therefore need to break them into smaller, and more generic pieces. Throughout the first year of VECMA, we have realized that this is possible, and we refer to these pieces as UQ Elements (or UQEs). Likewise, we realized that UQPs can be broken down, and therefore changed the nomenclature from Uncertainty Quantification Primitives to Uncertainty Quantification Patterns (allowing us to retain the UQP acronym).

Based on our current interpretation of UQPs, and our experiences in implementing UQ algorithms as part of our fast-track activities, we have identified a preliminary range of relevant UQEs, which are listed below. Please note that these definitions are likely to be revised as we obtain new insights:

- Parameter space specification: a definition of the parameter space in which uncertainty resides, including a distribution of probabilities that parameters have a specific value.
- Sampler: a component which, given a parameter space specification and probability distributions, generates random samples for a specific application.
- Encoder: a component that incorporates application-specific information in a generic definition of a UQ procedure (also called application mapping), enabling it to be carried out as a computation using e.g., a supercomputer.
- Decoder: a component that extracts generic UQ results from the (application-specific) output of a simulation run, enabling generic UQ tools to analyse the data and plan subsequent UQ activities.
- Collation: the process of aggregating the data generated by a range of executions into metrics that reflect the ensemble as a whole.
- Analysis: the process of extracting outcomes from collated data, in accordance with predefined specifications such as to allow for human interpretation or further workflow guidance.
- Probability distribution: a definition of the likelihood of a parameter assuming a particular value within its range of possible values.
- Worker: an independent task that serves to perform pre-instructed steps of a UQ procedure.

These elements help design and implementation activities, as many of them are essential to a range of UQPs. Consequently, we will be able, in many cases, to reuse existing code components not only when applying existing UQPs to new applications, but also when implementing new UQPs altogether.

2.4.2 Implementing UQPs

Within VECMA we focus largely on UQ algorithms that either leave a single-scale model implementation entirely untouched, or replace the implementation with a surrogate model, which can be either a stochastic approximation (trained through previous simulation work), or a cheaper and less accurate version of its original counterpart. As a result, a key component of implementing UQPs is to establish a generic mechanism for adding UQ *around* individual models. We provide an overview of such a mechanism in Figure 15, basically realizing UQP1:

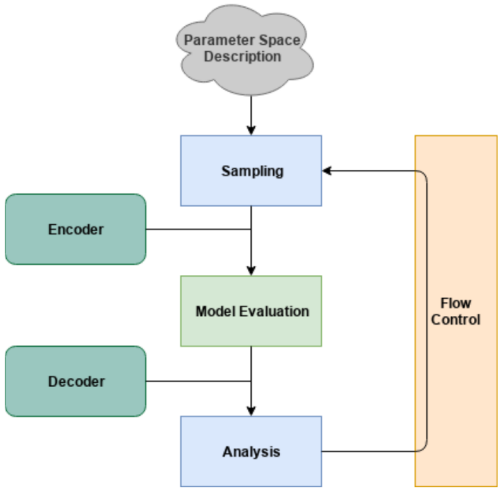

Figure 15: Generic implementation mechanism for performing UQ on a particular (single-scale) model.

In this mechanism we (a) define the parameter space in which we wish to do UQ, including parameter probability distributions, (b) draw samples from this space to create a *campaign* of runs, (c) encode the samples to translate them to definitions that can be executed as computations, (d) evaluate/execute the model implementation, (e) decode the results to extract the generic UQ output from the model, (f) analyse the UQ output to facilitate human interpretation and/or guide subsequent tasks, and (g) Refine and repeat steps b-f as needed, based on the results obtained in step f.

UQ, when performed in this way, is inherently a stochastic process both in a single- and multi-scale model context. As such, the exact sequence of tasks is not always known in advance, and the rules for controlling the flow of activities may rely on complex algorithms. To accommodate this in our implementation, we choose to implement UQPs as code-driven generic recipes as opposed to complex data structures. Although the exact implementation of these recipes may vary per tool, the use of Python code is most prevalent throughout the toolkit.

2.4.3 Key considerations for implementation and execution

For researchers to apply UQPs in a meaningful way, it is essential that their implementations are generically applicable, easy to use, and can be executed effectively on existing HPC platforms:

Generality: Implemented UQPs will only be of use to the full community if they are indeed application-agnostic. In VECMA we seek to preserve generality of UQPs in several ways. First, through the introduction of encoders and decoders in our toolkit (i.e., in EasyVVUQ) we isolate much of the application-specific information from the UQ logic. Second, we incorporate additional UQP functionalities (such as sampling and ensemble execution) at a low, application-agnostic level in our toolkit. Third, in our FabSim3 automation toolkit we use a plugin system for application- and domain-

specific functionalities, with plugins that rely on generic UQEs present elsewhere in the toolkit. In doing so we seek to encourage a separation of concerns, modularity, as well as re-use of predefined elements.

Ease of use: We seek to make implemented UQPs easy to use, and thereby promote uptake, as follows: First, we provide tutorials and example applications. For instance, the M12 release will feature at least three example applications. Second, we provide a range of tests, including automated unit and functional tests, peer-testing by developers, and thorough testing by internal and external alpha users every three months. Third, we provide active support to help new applications adopt the VECMA toolkit, and along with that the UQPs.

Execution: Within VECMA, many UQPs involve the intelligent sampling, resampling and coupling of single-scale models. If done naively, UQ across different models leads to an explosion in number of simulation evaluations, and even with advanced optimizations many applications will require thousands of jobs or more, while most supercomputer schedulers tend to support only $O(10)$ job executions at a given time. To make workflows with such large job counts tractable, we provide two mechanisms. First, we are able to quickly (and incrementally) generate very large numbers of job definitions using the sampling and encoding routines in EasyVVUQ. Second, we are currently incorporating a range of so-called *pilot job managers* within the VECMA toolkit (for instance, QCG Pilot Job Manager (QCG-PJM) and RADICAL Cybertools). These pilot job managers enable users to create a single container job on a remote supercomputer, and subsequently schedule 1000s of jobs or more within that job container. Due to the complex mechanisms that may be used to control the flow of UQ activities, we will require these pilot jobs to be controlled dynamically at runtime for advanced applications. Within the VECMA toolkit, we seek to establish this functionality, at scale, for the first time to our knowledge.

3 Conclusions and future work

WP2 is on track, and well underway to finalise task 2.2 as planned. Having established the main UQPs, and understanding how to break down the basic building block UQP1 into a number of generic building blocks (UQEs) and how to implement those in software, lays the groundwork for WP2. In close concertation with WP3 and WP4 the next steps will now be to further explore the UQPs as defined above, to test them in practice, to develop performance models and test those, and to look in more detail in the more advanced UQPs (UQP2-5). Moving into the next phase of VECMA also means that in WP2 we will now also start to investigate formal methods for validation and verification of multiscale models.

4 Annexes

4.1 The Multiscale Modelling and Simulation Framework

The MMSF provides an abstract way to understand the (computational) structure of multiscale models and multiscale simulations. Over the years, and in context of a series of EU funded projects, we have built strong confidence that the MMSF is an overarching framework that captures the characteristics of multiscale computing. Without going in much details here, we will highlight a few notions from the MMSF that will guide the definitions and design of UQPs. For full details we refer to [19,20] and references therein.

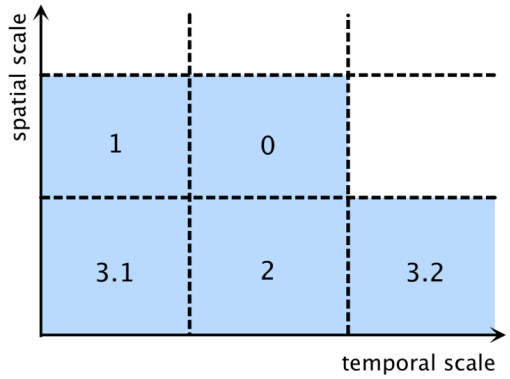

Figure 16: The interaction regions on the scale map, process A resides in region 0, and process B can then reside in 5 regions, leading to 5 types of interactions. 0 – scale overlap, Multiphysics; 1 – time scale separation; 2-spatial scale separation; 3.1 – classical micro-macro coupling; 3.2 – micro-macro coupling where a fast process on a large spatial scale is coupled to a slow process on a small spatial scale.

The first notion is the *Scale Separation Map* (SSM) and the associated *interaction regions* between two processes placed on the SSM, see Figure 16. Another relevant notion is the relation between the computational domains of two processes. These can either overlap (single domain) or be multi domain, where both computational domains exchange information through a boundary or small overlap region. Note that single-domain vs multi-domain is a property that is additional to the notion of interaction regions, that is for all interaction regions in Figure 16 one can find examples of single-domain or multi-domain multiscale applications. This has immediate consequences on how scale bridging information is exchanged between single scale models, and this notion will also have impact on UQPs. The notion of the interaction regions in combination with the relation between the computational domains leads to a powerful classification of multiscale systems.

Next we define a generic *Submodel Execution Loop* (SEL) that abstracts the computations in all single scale models as a while loop over three abstract operators (initialisation, a ‘solver’, and a boundary condition operator) and two operators that can observe the state of a single scale model (one inside the

while loop, and a second upon termination of a single scale model). We find that in coupling together single scale models in a multiscale model, only four *coupling templates*, defined as directed communication from an observation operator of one single scale model to a computing operator of another single scale model. Figure 17 shows the SEL of two processes, and an example of a coupling template in case the two processes would be time scale separated (so interaction region 1, 3.1, or 3.2). This coupling template is the *call-release* template.

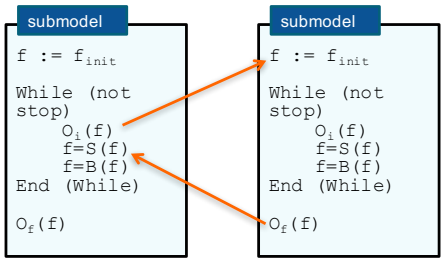

Figure 17: Example of two processes, interaction region 1, 3.1, or 3.2, showing the SEL and the coupling templates (in this case the call – release pair).

Next, we introduce the notion of multiscale computing, and the two main multiscale computing paradigms, acyclic (or loosely coupled, or workflows) and cyclic (or tightly coupled), see Figure 18. In acyclic multiscale computing one single scale model provides input to another, and single scale models are executed once. This can be seen as a traditional workflow, with the difference that the scale bridging, the arrow between the single scale models could entail a quite complicated hand-shake. In general, the fact that these models operate on different scales means that the very nature of the models may be radically different – for example, one might be particulate/stochastic, the other continuum based and deterministic. Getting a "handshake" between both models is frequently complicated and requires multiple steps and computations. In cyclic multiscale computing, single scale models call each other in an iterative loop, and therefore single scale models can execute many times. For such cyclic computing dedicated coupling libraries are required.

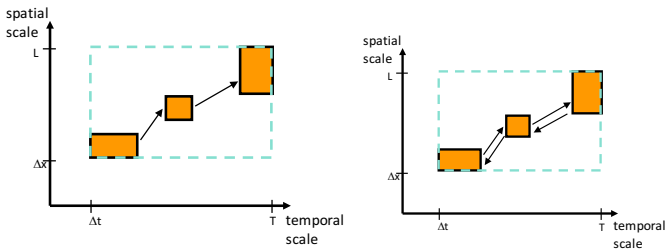

Figure 18: Acyclic (left) and cyclic (right) multiscale applications

Finally we need to specify how many instances of single scale models are executed, if this number is fixed or dynamic, and in case of cyclic applications, how many synchronization points are required (so

how many cycles are passed in the cyclic application) and if the number of synchronization points are static or dynamic. All possible combinations lead to 9 different *coupling topologies*, see Figure 19.

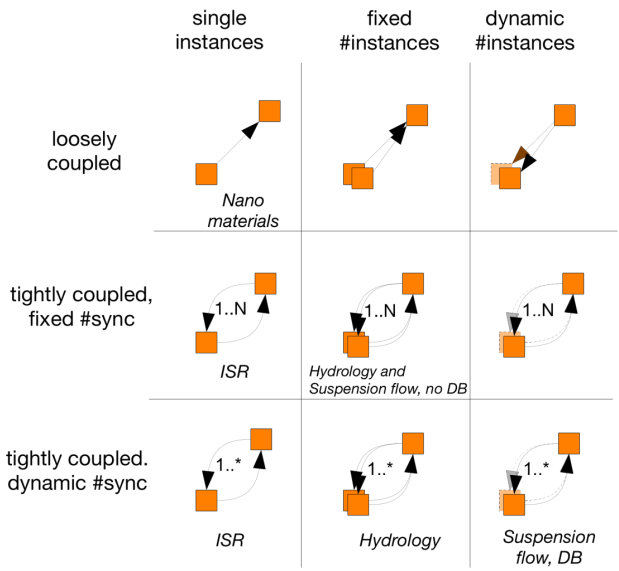

Figure 19: Coupling topologies

The *Multiscale Modelling Language* (MML) translates all these concepts into a graphical (gMML) and machine readable (xMML) specification of the multiscale model that contains in principle sufficient information for execution of the multiscale model in any type of computing environment. In the earlier MAPPER and COMPAT projects we have demonstrated all these capabilities in the context of *Distributed Multiscale Computing* (DMC) and *High Performance Multiscale Computing* (HPMC).

Another relevant feature of the MMSF that may be of relevance to UQPs is the notion of task graphs for multiscale computing. Borgdorff et al. already introduced task graphs when specifying the foundations of the MMSF [19]. As shown in Figure 20, a task graph can be derived from an xMML specification of a multiscale application, and the task graph in turn can be used as input for scheduling software. We have demonstrated that task graphs can automatically be derived from xMML [19] and demonstrated the use of task graphs for one specific application [27].

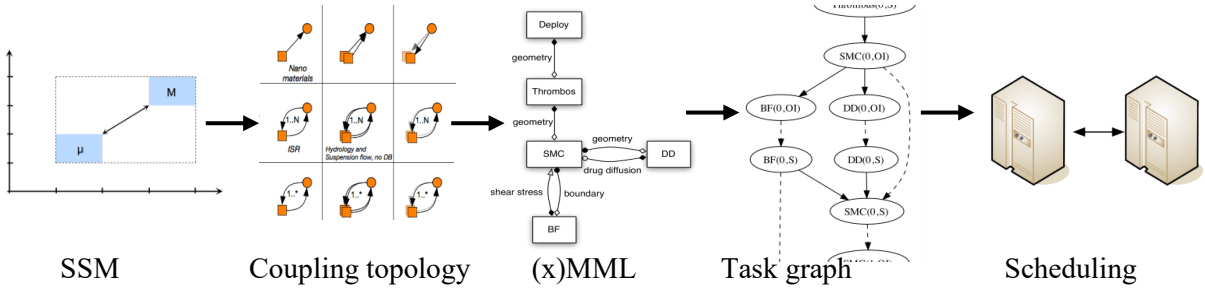

Figure 20: Several stages of description of a multiscale model in the MMSF, starting from the Scale Separation Map, details of the Coupling Topology are added, followed by a full specification in terms of xMML, from which a Task Graph is derived that can then be used as input to scheduling software.

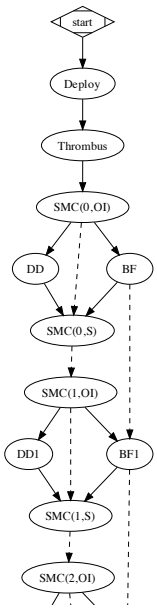

Figure 21: example of a task graph, showing the initialisation and first two cycles in the ISR3D model. Note that in the ISR application typically a few thousands of full cycles are performed.

Task graphs were introduced in the MMSF for deadlock detection, validity checking, and for estimating computational costs and scheduling. An example of a task graph for the In-Stent Restenosis application (see deliverable D4.1) is shown in Figure 22.

A task graph is a directed acyclic graph of tasks (the nodes) and their dependencies or data flows (the edges). It can be used for scheduling on parallel and/or distributed computing resources [32] and in the context of VECMA, to apply UQPs on the task graph. It can also be seen as a serialized or unfolded graph of the MML description, which may be cyclic. Task graphs can get extremely large, growing exponentially in the number of temporally scale separated submodels, exacerbated by submodels that have a lot of iterations. A methods to reduce the number of nodes is collapsing redundant nodes, which is also demonstrated in [19]. In fact, Figure 22 shows such reduced graph.

4.2 Examples of applying semi-intrusive multiscale UQ algorithms

4.2.1 1D reaction-diffusion system

The first case study is a 1D reaction-diffusion model with slow diffusion and fast reaction [10]. The response of this system consists of two two-dimensional fields, which we name *u* and *v*. The UQ result obtained with the MC of the final time step is presented in Figure 22. The results of the mean value for

u and *v* are approximately reversed. At the same time, the standard deviations of *u* and *v* have a similar pattern. However, since the maximum value in space of *u* is much greater than *v*, the relative uncertainty of *v* represented by the coefficient of variation reaches 100% at some locations, where this value for *u* is about 36%. The execution time for the tested multiscale UQ algorithms is presented in Figure 23.

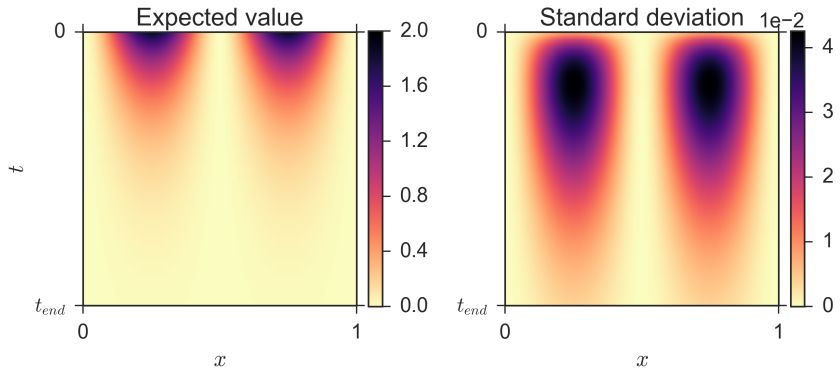

Figure 22: The expected value and the standard deviation of the two systems estimated by the Monte Carlo method

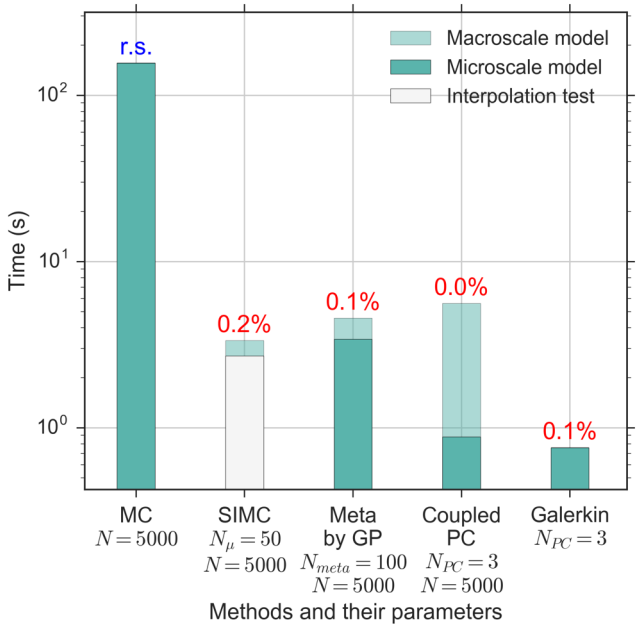

Figure 23: Comparison of UP methods in terms of execution time. Here and later, N_{meta} is the number of samples used to build the data-driven metamodel, N_{PC} is the truncated power in the Polynomial Chaos method, and "r.s" denotes the reference solution. The numbers above the bars are the mean relative errors in the results of the standard deviation obtained by the methods versus the MC results. The execution time broken down into time spent in respectively the macroscale model, the microscale model, and the interpolation test (only for the SIMC).

4.2.2 Gray-Scott model

The second example is a two-dimensional Gray-Scott model as presented in [10]. The UQ result obtained with the MC of the final time step is presented in Figure 24. The results of the mean value are still quite close to the patterns from Figure 24, and this results for *u* and *v* are approximately reversed. At the same time, the standard deviations of *u* and *v* have a similar pattern. However, since the maximum value in space of *u* is much greater than *v*, the relative uncertainty of *v* represented by the coefficient of variation reaches 100% at some locations, where this value for *u* is about 36%.

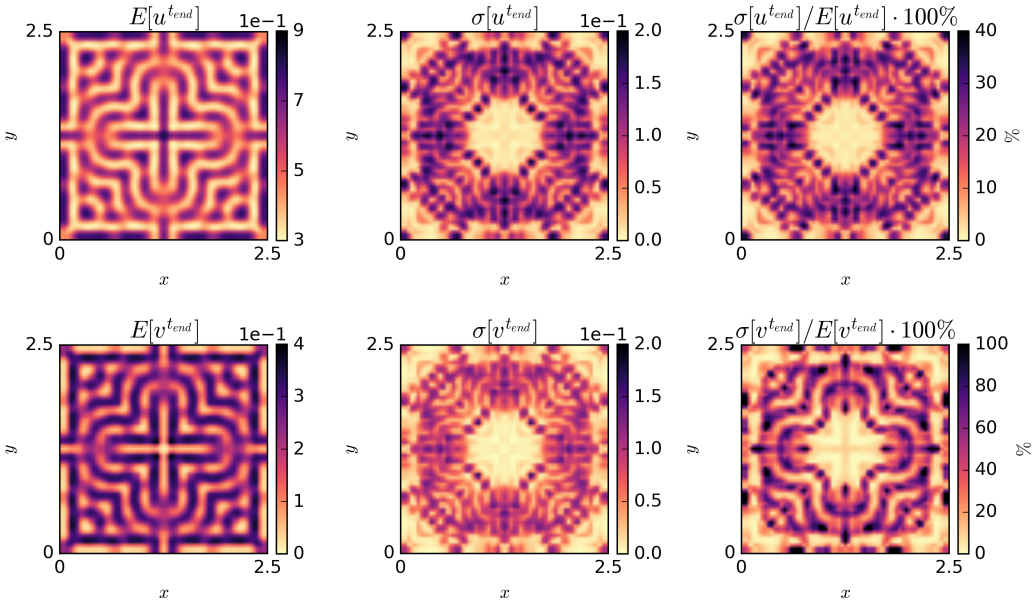

Figure 24: Uncertainty estimation result obtained by the MC method: the mean value (left column), the standard deviation (central column) and the coefficient of variation (right column) of the concentration *u* (upper row) and *v* (bottom row).

A comparison of the computational time and the error in the standard deviation by several semi-intrusive multiscale UQ methods is presented in Figure 25, where the MC result is used as a reference solution. The SIMC and metamodeling with GP result in a significant drop in the execution time while the coupled intrusive and non-intrusive PC and the Galerkin methods are computationally more expensive than the MC method. Moreover, the error in the results of the last two methods exceeds 60%. The high value of the error is due to the nonlinear nature of the Gray-Scott model, which cannot be approximated by a series of low order polynomials. The results obtained by the SIMC and the metamodeling with the GP are much closer to the MC results. In this example, the interpolation test in the SIMC is not passed, and, therefore, 50 samples are used to compute uncertainty with the MC, which produces a 7.7% error instead of 11% when the result of the SIMC is accepted.

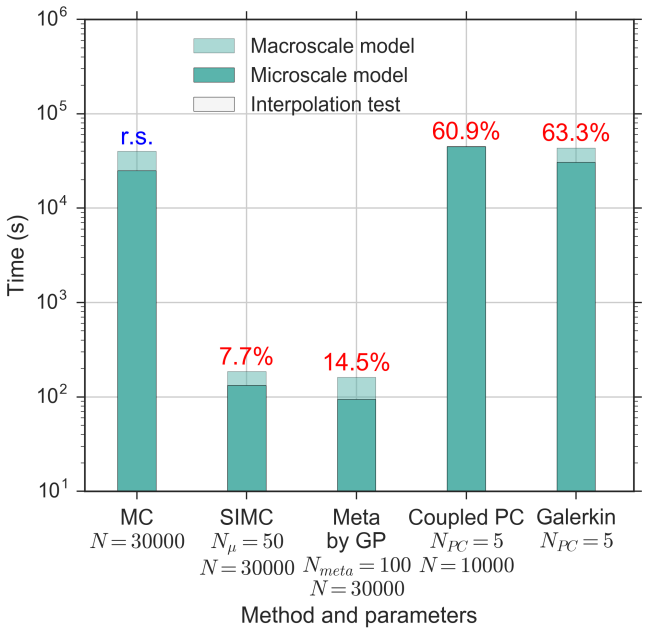

Figure 25: Comparison of the performance of the UQ methods applied to the Gray-Scott model, where "r.s" denotes the reference solution, and the percentage indications above the columns are the mean relative error in the estimates of the standard deviation.

4.2.3 ISR2D

The next application is a 2-dimensional multiscale model of in-stent restenosis (ISR2D) [22]. Figure 26a shows the estimated mean value obtained by four UQ methods that were tested: the quasi-Monte Carlo (QMC) and the semi-intrusive multiscale methods, where two metamodels are obtained with a data-driven method (DD Meta I and DD Meta II) and one by simplified physics (Phy Meta). The 95% bootstrap confidence interval is plotted as a red shaded area around the results from the QMC method, as presented in [23]. All three metamodeling results result in a statistically significant underestimation of the mean value (two-valued t-test, $p < 0.01$).

ISR2D is subject to both epistemic and aleatory uncertainty. Some model inputs are uncertain due lack of knowledge, and the model is also stochastic itself because it simulates the natural variability of the process of interest. We estimated the total standard deviation, as well as an upper bound of the partial standard deviation due to aleatory model uncertainty. As Figure 26b shows, the metamodeling estimates are high, but for the data-driven metamodels still within the 95% confidence interval of the QMC result.

The coefficients of variation (CoV) shown in Figure 26c are equal to the ratio between the (partial) standard deviation and the mean value, and are a measure of the relative model output uncertainties. Here we observe that the underestimation of the mean value leads to the overestimation of the CoV.

Figure 26d-g show the distribution of the cross-sectional areas at the final simulation time step. The dashed vertical line indicates a restenosis threshold, defined as 50% occlusion of the original lumen area. Thus, about 9.7% out of all samples obtained by QMC result in restenosis. The data-driven metamodel results are about half this value. The results with the physical metamodel shows result of only 1.8%, and a shift of the probability density function to the left is visible.

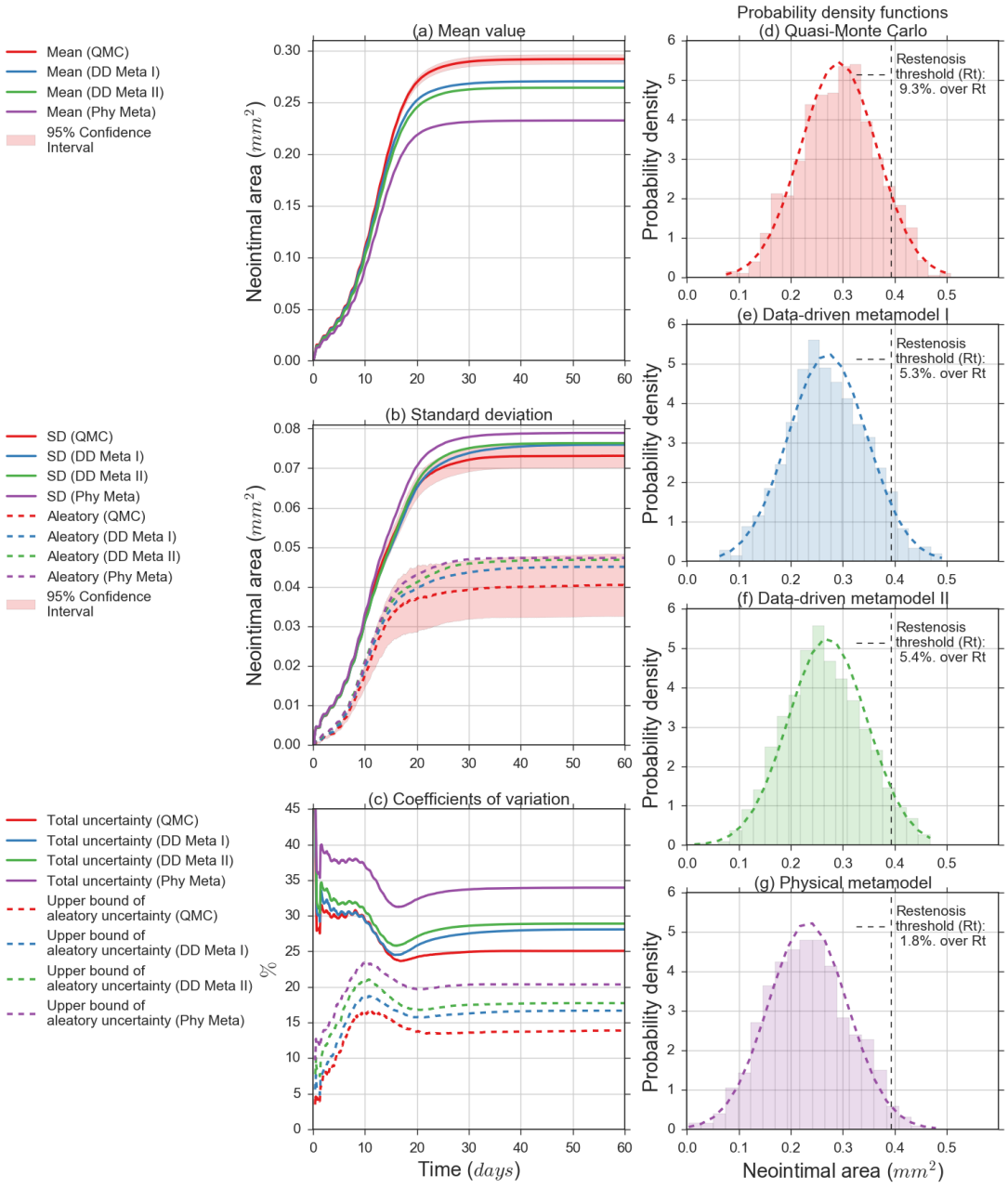

Figure 26: Analysis of the uncertainty measures by the four UQ methods: the quasi-Monte Carlo (QMC) method, metamodelling methods by data-driven (DD meta I and II) approach and by simplified physics (phys meta).

Figure 27 shows the time needed for a single model run using the four UQ methods: the quasi-Monte Carlo (QMC), the two different data-driven metamodels (DD Meta I and II), and the metamodeling by the simplified physical surrogate. The light colors indicate the total execution time of the whole multiscale model, and the dark colors indicate the portion of the time taken by the micro model or surrogate. For the metamodeling methods, the indicated time includes the construction of the metamodels.

DD Meta I reduces total runtime by almost half. DD Meta II, which uses less data, is faster still, at five times the speed of the original. The physical metamodel is fastest, but only by a small margin over DD Meta II.

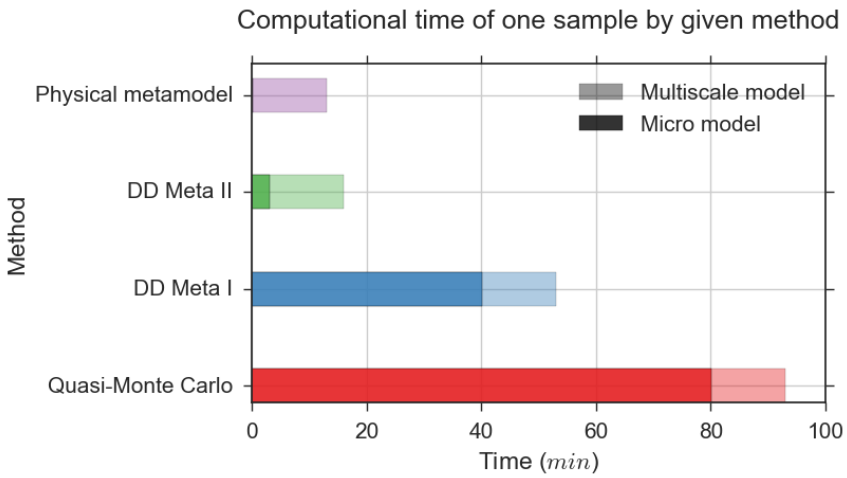

Figure 27: Comparison of the computational time per one model run with different UQ methods.

5 References

[1] A.G. Hoekstra, B. Chopard, D. Coster, S. Portegies Zwart, P. V. Coveney, Multiscale computing for science and engineering in the era of exascale performance, Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. 377 (2019) 20180144. doi:10.1098/rsta.2018.0144.

[2] S. Alowayyed, D. Groen, P. V. Coveney, A.G. Hoekstra, Multiscale Computing in the Exascale Era, J. Comput. Sci. 22 (2017) 15–25. doi:10.1016/j.jocs.2017.07.004.

[3] A.G. Hoekstra, B. Chopard, P. V. Coveney, Multiscale modelling and simulation: a position paper, Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. 372 (2014) 20130377. doi:10.1098/rsta.2013.0377.

[4] P.M.A. Sloot, A.G. Hoekstra, Multi-scale modelling in computational biomedicine, Br. Bioinform. 11 (2010) 142–152. doi:10.1093/bib/bbp038.

[5] S. Karabasov, D. Nerukh, A.G. Hoekstra, B. Chopard, P. V. Coveney, Multiscale modelling: approaches and challenges, Philos. Trans. R. Soc. A. 372 (2014) 20130390. doi:10.1098/rsta.2013.0390.

[6] R.H. Johnstone, E.T.Y. Chang, R. Bardenet, T.P. de Boer, D.J. Gavaghan, P. Pathmanathan, et al., Uncertainty and variability in models of the cardiac action potential: Can we build trustworthy models?, J. Mol. Cell. Cardiol. 96 (2016) 49–62. doi:10.1016/J.YJMCC.2015.11.018.

[7] R.C. Smith, Uncertainty quantification: theory, implementation, and applications, SIAM, 2013.

[8] O.P. Le Maître, O.M. Knio, Spectral Methods for Uncertainty Quantification, Springer Netherlands, Dordrecht, 2010. doi:10.1007/978-90-481-3520-2.

[9] X. Wan, G.E. Karniadakis, An adaptive multi-element generalized polynomial chaos method for stochastic differential equations, J. Comput. Phys. 209 (2005) 617–642. doi:10.1016/J.JCP.2005.03.023.

[10] Nikishova, Anna, A.G. Hoekstra, Semi-intrusive uncertainty quantification for multiscale models, ArXiv Prepr. (2018) 1806.09341.
[11] B. Wang, T. Chen, Gaussian process regression with multiple response variables, Chemom. Intell. Lab. Syst. 142 (2015) 159–165. doi:10.1016/J.CHEMOLAB.2015.01.016.
[12] O.K. Oyebamiji, D.J. Wilkinson, P.G. Jayathilake, T.P. Curtis, S.P. Rushton, B. Li, et al., Gaussian process emulation of an individual-based model simulation of microbial communities, J. Comput. Sci. 22 (2017) 69–84. doi:10.1016/J.JOCS.2017.08.006.
[13] T. Zhan, L. Fang, Y. Xu, Prediction of thermal boundary resistance by the machine learning method, Sci. Rep. 7 (2017) 7109. doi:10.1038/s41598-017-07150-7.
[14] Y. Liu, J. Guo, Q. Wang, D. Huang, Prediction of Filamentous Sludge Bulking using a State-based Gaussian Processes Regression Model, Sci. Rep. 6 (2016) 31303. doi:10.1038/srep31303.
[15] M. Gerritsma, J.-B. van der Steen, P. Vos, G. Karniadakis, Time-dependent generalized polynomial chaos, J. Comput. Phys. 229 (2010) 8333–8363. doi:10.1016/J.JCP.2010.07.020.
[16] R. Archibald, M. Chakoumakos, T. Zhuang, Characterizing the elements of Earth's radiative budget: Applying uncertainty quantification to the CESM, Procedia Comput. Sci. 9 (2012) 1014–1020. doi:10.1016/J.PROCS.2012.04.109.
[17] J. Feinberg, H.P. Langtangen, Chaospy: An open source tool for designing methods of uncertainty quantification, J. Comput. Sci. 11 (2015) 46–57. doi:10.1016/J.JOCS.2015.08.008.
[18] N.E. Owen, P. Challenor, P.P. Menon, S. Bennani, Comparison of Surrogate-Based Uncertainty Quantification Methods for Computationally Expensive Simulators, SIAM/ASA J. Uncertain. Quantif. 5 (2017) 403–435. doi:10.1137/15M1046812.
[19] J. Borgdorff, J.-L. Falcone, E. Lorenz, C. Bona-Casas, B. Chopard, A.G. Hoekstra, Foundations of distributed multiscale computing: Formalization, specification, and analysis, J. Parallel Distrib. Comput. 73 (2013) 465–483. doi:http://dx.doi.org/10.1016/j.jpdc.2012.12.011.
[20] B. Chopard, J. Borgdorff, A.G. Hoekstra, A framework for multi-scale modelling, Philos. Trans. R. Soc. A. 372 (2014) 20130378. doi:10.1098/rsta.2013.0378.
[21] B. Chopard, J.-L. Falcone, P. Kunzli, L. Veen, A. Hoekstra, Multiscale modeling: recent progress and open questions, Multiscale Multidiscip. Model. Exp. Des. 1 (2018) 57–68. doi:10.1007/s41939-017-0006-4.
[22] A. Nikishova, L. Veen, P. Zun, A.G. Hoekstra, Semi-intrusive multiscale metamodelling uncertainty quantification with application to a model of in-stent restenosis, Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. 377 (2019) 20180154. doi:10.1098/rsta.2018.0154.
[23] A. Nikishova, L. Veen, P. Zun, A.G. Hoekstra, Uncertainty Quantification of a Multiscale Model for In-Stent Restenosis, Cardiovasc. Eng. Technol. (2018) 1–14. doi:10.1007/s13239-018-00372-4.
[24] J.-L.J.-L. Falcone, B. Chopard, A. Hoekstra, MML: towards a Multiscale Modeling Language, Procedia Comput. Sci. 1 (2010) 819–826. doi:DOI: 10.1016/j.procs.2010.04.089.
[25] J. Borgdorff, E. Lorenz, A.G. Hoekstra, J. Falcone, B. Chopard, A Principled Approach to Distributed Multiscale Computing, from Formalization to Execution, in: E-Science Work. (EScienceW), 2011 IEEE Seventh Int. Conf., 2011: pp. 97–104. doi:10.1109/eScienceW.2011.9.
[26] J. Borgdorff, M. Mamonski, B. Bosak, K. Kurowski, M. Ben Belgacem, B. Chopard, et al., Distributed multiscale computing with MUSCLE 2, the Multiscale Coupling Library and Environment, J. Comput. Sci. 5 (2014) 719–731. doi:http://dx.doi.org/10.1016/j.jocs.2014.04.004.
[27] J. Borgdorff, C. Bona-Casas, M. Mamonski, K. Kurowski, T. Piontek, B. Bosak, et al., A Distributed Multiscale Computation of a Tightly Coupled Model Using the Multiscale Modeling Language, Procedia Comput. Sci. 9 (2012) 596–605. doi:10.1016/j.procs.2012.04.064.
[28] J. Borgdorff, M. Ben Belgacem, C. Bona-Casas, L. Fazendeiro, D. Groen, O. Hoenen, et al., Performance of distributed multiscale simulations, Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. 372 (2014) 20130407. doi:10.1098/rsta.2013.0407.
[29] D. Groen, J. Borgdorff, C. Bona-Casas, J. Hetherington, R.W. Nash, S.J. Zasada, et al., Flexible composition and execution of high performance, high fidelity multiscale biomedical simulations, Interface Focus. 3 (2013) 20120087. doi:10.1098/rsfs.2012.0087.
[30] M.B. Belgacem, B. Chopard, J. Borgdorff, M. Mamonski, K. Rycerz, D. Harezlak, Distributed multiscale computations using the MAPPER framework, Procedia Comput. Sci. 18 (2013) 1106–1115.
[31] S.A. Alowayyed, Patterns for multiscale computing, University of Amsterdam, 2018.
[32] Y.-K. Kwok, I. Ahmad, Benchmarking and Comparison of the Task Graph Scheduling Algorithms, J. Parallel Distrib. Comput. 59 (1999) 381–422. doi:10.1006/jpdc.1999.1578.