D3.1: Report on the fast-track implementation of the VECMA toolkit

Due Date	14 March 2019
Delivery	6 March 2019
Submission of updated version	N/A
Lead Partner	Brunel University London
Dissemination Level	Public
Status	Final
Approved	Executive Board
Version	V1.0

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
06/02/2019 v0.1	Hamid Arabnejad	First version based on ICCS paper.
11/02/2019 v0.2	Derek Groen	Heavily revised version, with all sections complete.
4/03/2019 v0.4	Hamid Arabnejad	Incorporated first reviewer comments.
5/03/2019 v0.5	Derek Groen	All comments incorporated.
5/03/2019 v0.8	Peter Coveney	Provided final comments.
5/03/2019 v1.0	Derek Groen	All comments incorporated, version to be submitted.

CONTRIBUTORS

- Derek Groen, UBRU - Author
- Hamid Arabnejad, UBRU - Author
- Vytautas Jancauskas, LRZ - Contributor
- Robin Richardson, UCL - Contributor
- Robbie Sinclair, UCL - Contributor
- David Wright, UCL - Contributor
- Maxime Vassaux, UCL - Contributor
- Paul Karlshoefer, BULL, - Contributor
- Peter Coveney, UCL - Contributor
- Xuanye Gu, UCL - Contributor
- Apostolos Evangelopoulous, UCL - Contributor
- Tomasz Piontek, PSNC - Contributor
- Piotr Kopta, PSNC - Contributor

Disclaimer

Definition and Acronyms

Acronym	Definition
VVUQ	Verification Validation and Uncertainty Quantification
AUC	Acceptance Use Case
UQP	Uncertainty Quantification and sensitivity analysis Patterns
VVP	Verification and Validation Patterns
HPC	High Performance Computing
QCG	Quality in Cloud and Grids
VECMAtk	Toolkit for Verified Exascale Computing for Multiscale Applications
EasyVVUQ	A tool for constructing and performing Verification Validation and Uncertainty Quantification procedures, intended to be easy to use.

TABLE OF CONTENTS

1 Executive summary

- This document reports on the fast-track implementation of the VECMA toolkit (VECMAtk), which is being developed using an evolutionary prototyping approach.
- We have already made three internal releases (M1, M3 and M6), and successfully applied two components (FabSim3 and EasyVVUQ) to multiple diverse HPC applications. Both components are now in *open development*, i.e. they can be checked, modified and used by anyone at any time.
- We are on track to deliver the features proposed for the major releases (M12, M24, M30).
- We introduced a growing group of 'alpha users', which allows us to gain feedback during the early stages of the project. Three alpha users gave feedback on the M3 release, five users on the M6 release.
- We present the proposed release schedule of the toolkit for the VECMA project, which also includes minor releases at every three months, to be reviewed by alpha users.
- Integration with the QCG middleware is forthcoming in the M9 release, combined with the incorporation of additional application domains.
- Not in the proposal, but important for uptake, is the fact that we have established continuous integration testing for the toolkit, and created a containerised version for easier distribution.

2 The fast-track implementation of the VECMA toolkit

This deliverable reports on the fast-track implementation of the VECMA toolkit (VECMAtk). We summarise the current status of the technical development, list the anticipated features for the fast-track release (M12), and provide a provisional list of features for the deep-track release (M24). This deliverable is part of VECMA's Work Package 3, in which we are creating a software toolkit to enable automated Verification, Validation and Uncertainty Quantification (VVUQ) for multiscale applications that can be deployed on emerging exascale platforms. A detailed description of the VECMA software stack as a whole is given in D5.1 and a systematic overview of the VECMA applications in D4.1. Therefore, we will only provide material on these topics where strictly necessary within this deliverable. In terms of cross-WP connectivity, the toolkit integrates the primitives developed in WP2 into widely adopted software for multiscale computing, and is positioned to support the applications developed in WP4 (as well as others), and to make use of the infrastructure provisioned in WP5 (as well as other resources). The toolkit is one of the major outcomes of the VECMA project, and the three

major releases (M12, M24 and M30) will be accompanied with wide-ranging dissemination activities (WP6).

2.1 Introduction

The overarching goal of computational modelling is to provide insight into questions that in the past could only be addressed by resource-intensive experimentation, if at all. In order for the results of computational science to impact decision making outside of basic science, for example in industrial or clinical settings, it is vital that they are accompanied by a robust understanding of their degree of certainty. In practice this can be decomposed into checks of whether the codes employed are solving the correct equations (*validation*), in an accurate manner (*verification*), and the provision of error bars (*uncertainty quantification*). These processes, collectively known as VVUQ, provide the basis for determining our level of trust in any given model and the results obtained using it. Recent advances in the available computational resources and algorithms designed to exploit it mean that it is increasingly possible to conduct the additional sampling required by VVUQ even for highly complex calculations and workflows.

The main objective of VECMA is to produce a toolkit (VECMAtk) which automates VVUQ for high performance (multiscale) computing applications as much as possible, supporting execution on petascale and emerging exascale environments. The toolkit will flexibly support a wide range of tools and middleware, such as FabSim3, QCG, and RADICAL-Cybertools, to facilitate the process of design and execution of applications on HPC infrastructure. VECMAtk will vastly reduce the time and effort invested by researchers to render their multiscale simulation output “actionable”, i.e. trusted and reliable enough to include in key decision-making processes, with particular emphasis on exploiting present and future (exascale) supercomputing power. It will improve simulation accuracy and reliability by providing tools to access their uncertainties. In addition, VECMAtk will allow VVUQ practices to thrive via the widespread exploitation of resources in High Performance Computing (HPC), whose power is becoming greater as we approach the era of exascale supercomputing over the next few years. Currently, we consider a wide range of applications, from fusion and advanced materials through climate and forced population displacement, to drug discovery and personalised medicine, to test and validate the toolkit. However, during the lifetime of the project, we aim to test and support our approach for a wider range of scientific and social scientific applications.

Multiscale modelling presents particular difficulties as applications in this field frequently consist of complex workflows where uncertainty propagates through highly varied components, which may only

be executed conditionally. Additionally, uncertainties may be associated with the process of transforming the output variables from one scale to another, as information is necessarily lost when coarsening a model and interpolation (or even the "anti-thermodynamic" activity of creating information (Delgado-Buscaloni, 2003)) is necessary when moving to a more fine-grained representation. Within VECMA, we wish to add VVUQ to existing scientific workflows without changing researchers' conception of the problem they are investigating. This goal informs the conception of the toolkit which is designed such that it:

- is highly modular
- has minimal installation requirements
- provides control over where (locally or on remote resources) and at what time analysis takes place

This means that VECMAtk is a collection of elements that can be reused and recombined in different workflows. Our aim is to define stable interfaces, data formats and APIs that facilitate VVUQ in the widest range of applications. This toolkit will allow users in the application domains to combine generic and lightweight primitives to create a system-specific VVUQ approach that will extend to the exascale and cover all relevant space and time scales, including the scale-bridges between them.

2.2 WP3 work scheme

Most of the VECMAtk development takes place as part of WP3. In the development of VECMAtk, we categorize development into fast-track development, which treats underlying applications as a black box (*non-intrusive*), and deep-track development, which takes into account the coupling mechanisms between single models (*semi-intrusive*) or even the algorithms of the single scale models themselves (*fully intrusive*). In addition, we concentrate our development activities around three essential releases. The initial (fast-track) release (M12), the preliminary deep-track release (M24), and the full release (M30). After M30, our efforts will focus on bolstering and future-proofing the toolkit, as well as increasing its uptake among the wider applications communities.

2.2.1 Development teams

Within work package 3 we use a work approach that differs from other work packages. Meetings across the work package are held on only an occasional basis. Instead, we have divided activities in the work package across several smaller *development teams*. Each development team is temporary, focused on a particular priority in the project, and consists of 2 to 7 members. One member coordinates the development team, and liaises with the technical coordinator (Derek Groen) to ensure that development is properly aligned with the other teams. Members and partners are free to join

and leave development teams as they see fit, and as a result these teams may be split (if too large) or disbanded (if too small).

2.3 General overview of the VECMA toolkit

Given the objectives of the project and WP3, the main factors that shape the development approach of VECMAtk are:

- the need to fit into existing applications with minimal modification effort,
- the wide range of target application domains,
- the flexible and recombinable nature of the toolkit itself, and
- the geographically distributed nature of the users, and especially the developers.

To support these needs and characteristics, we have chosen to adopt an evolutionary prototyping approach (See Figure 1). In this approach, existing application developers initially establish VVUQ techniques using their own scripts, which they share with VECMAtk developers together with additional needs that they have not been able to easily address themselves. These scripts and requirements, together with existing libraries, are called *Acceptance Use Cases* (AUCs), and form the base from which the development team develops initial prototypes. These prototypes are then periodically tested and refined, with the user feedback and integration testing guiding further developments. As a result, some prototypes are reduced in scope, simplified, or removed altogether; and some prototypes are being refined into more advanced, flexible, scalable or robust tools.

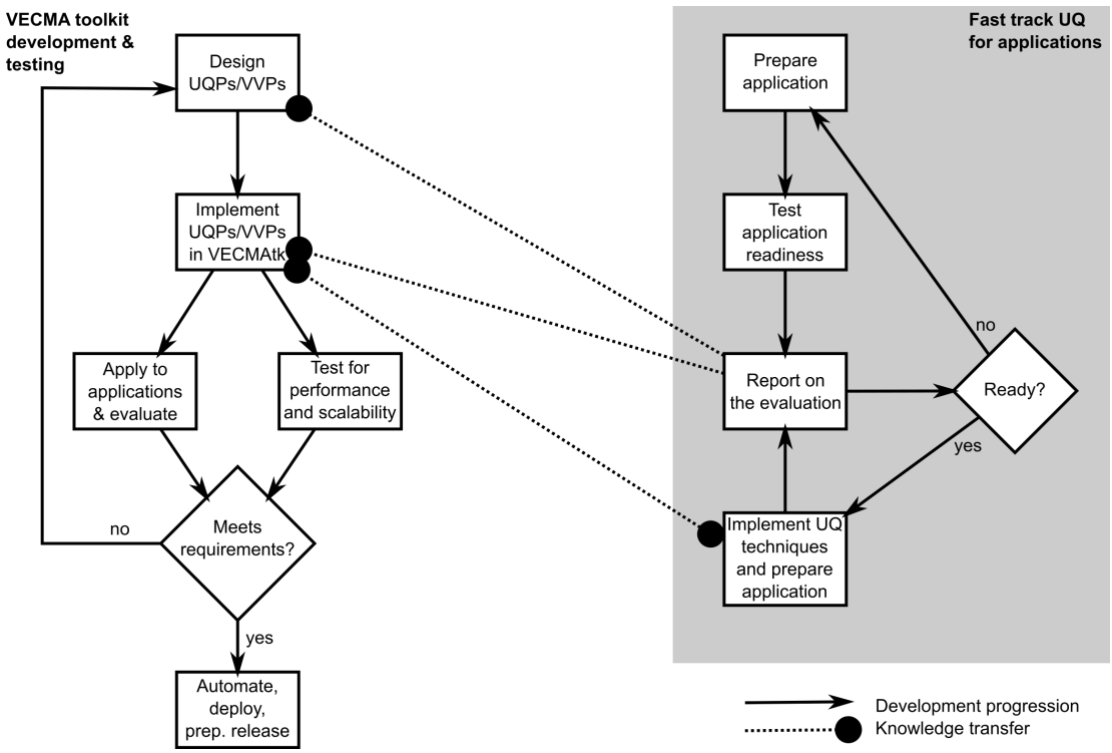

Figure 1: Graphical overview of the evolutionary prototyping approach.

As part of our prototyping process, we identify common workflow patterns and software elements (for example those needed to encode complex parameter distributions) that can be abstracted for re-use in a wide range of application scenarios. We label patterns found in verification and validation contexts *VVPs*, and those associated with uncertainty quantification *UQPs*. The definition of a UQP of VVP should never require the use of any specific execution management platform but the toolkit is envisioned to provide multiple solutions that facilitate workflows including diverse sampling algorithms and job types running on heterogeneous resources. In the next section we provide a brief impression of how VECMAtk operates, and how its different parts either work together now, or are expected to do so by the M12 release. A more detailed overview of the VECMA architecture as a whole can be found in D5.1.

2.3.1 Characterizing VECMAtk

VECMAtk currently consists of three components: EasyVVUQ, FabSim3, and the QCG software environment. We use EasyVVUQ to define VVUQ procedures for the (multiscale) application. The VVUQ procedure includes four main activities, (a) sampling, (b) simulation execution, (c) result aggregation, and (d) analysis. Subsequently, we use FabSim3 to construct an automated workflow which combines the procedures from VVUQ with application- and user-specific information, as well as definitions of which tasks are required to perform the simulations themselves. Although FabSim3 supports basic job submission and small ensemble runs directly to individual machines using SSH and GSISSH, we can exploit QCG-Client and QCG-Broker to enable the efficient scheduling, and execution of applications across resources. An example workflow which involves all the current VECMAtk components can found in Figure 2.

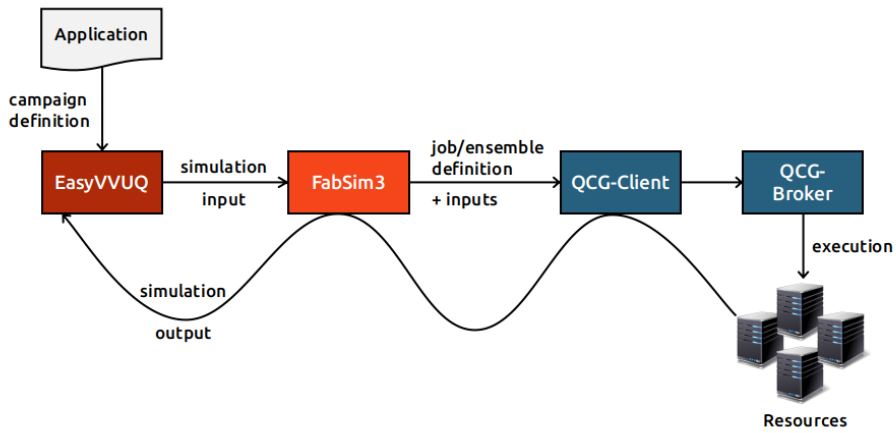

Figure 2: An example workflow where all the existing components of VECMAtk are used. Note that components can be flexibly combined, and many workflows require only a subset of the full toolkit.

EasyVVUQ is a Python library designed to simplify the implementation of (primarily blackbox) VVUQ workflows involving existing applications. The library is designed around a breakdown of such workflows into four distinct stages; sampling, simulation execution, result aggregation, and analysis. The execution step is deemed beyond the remit of the package (it can be handled for instance by FabSim3 or QCG Client), whilst the other three stages are handled separately. A common data structure, the *Campaign*, which contains information on the application being analysed alongside the runs mandated by the sampling algorithm being employed, is used to transfer information between each stage. The architecture of EasyVVUQ is shown in Figure 3.

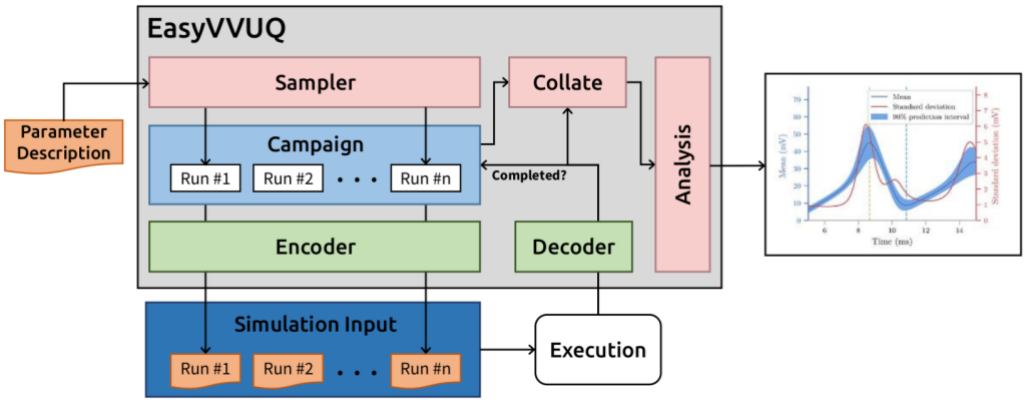

Figure 3: Overview of the EasyVVUQ Architecture.

The user provides a description of the model parameters and how they might vary in the sampling phase of the VVUQ pattern, for example specifying the distribution from which they should be drawn and physically acceptable limits on their value. This is used to define a *Sampler* which populates the Campaign with a set of run specifications based on the parameter description provided by the user. The Sampler may employ one of a range of algorithms such as the Monte Carlo or Quasi Monte Carlo approaches (Sobol, 1998). At this point all of the information is generic in the sense that it is not specific to any application or workflow. The role of the *Encoder* is to create input files which can be used in a specific application. Included in the base application is a simple templating system in which values from the Campaign are substituted into a text input file. For many applications it is envisioned that specific encoders will be needed and the framework of EasyVVUQ means that any class derived from a generic Encoder base class is picked up and may be used. This enables EasyVVUQ to be easily extended for new applications by experienced users. The simulation input is then used externally to the library to execute the simulations. The role of the *Decoder* is twofold, to record simulation completion in the Campaign and to extract the output information from the simulation runs. Similarly

to the Encoder, the Decoder is designed to be user extendable to facilitate analysis of a wide range of applications. The Decoder is used in the collation step to provide a combined and generic expression of the simulation output for further analysis (for example the default is to bring together output from all simulation runs in a Pandas dataframe). Following the output collation, we provide a range of analysis algorithms which produce the final output data. Whilst the library was originally designed for acyclic `blackbox' VVUQ workflows development is ongoing to allow the library to be used in more complex patterns. The combination of different UQPs and VVPs in one application also leads to a cognitively complex workflow structure, where different sets of replicas need to be constructed, organised, executed, analysed, and actioned upon (i.e. triggering subsequent execution and/or analysis activities).

FabSim3 (Groen, 2016) (http://www.github.com/djgroen/FabSim3) is a freely available tool that supports the construction, creation and execution of these complex workflows, and allows users to invoke them on remote resources using one-liner commands. In contrast to its direct predecessor FabSim, FabSim3 inherently supports the execution of job ensembles, and provides a plug-in system which allows users to customize the toolkit in a modular and lightweight manner (e.g., evidenced by the minimalist open-source FabDummy plugin [http://www.github.com/djgroen/FabDummy]). We provide an overview of the FabSim3 architecture in Figure 4. In the context of VECMA, FabSim3 plays a key role in introducing application-specific information in the Execution Layer, and in conveniently combining different UQPs and VVPs.

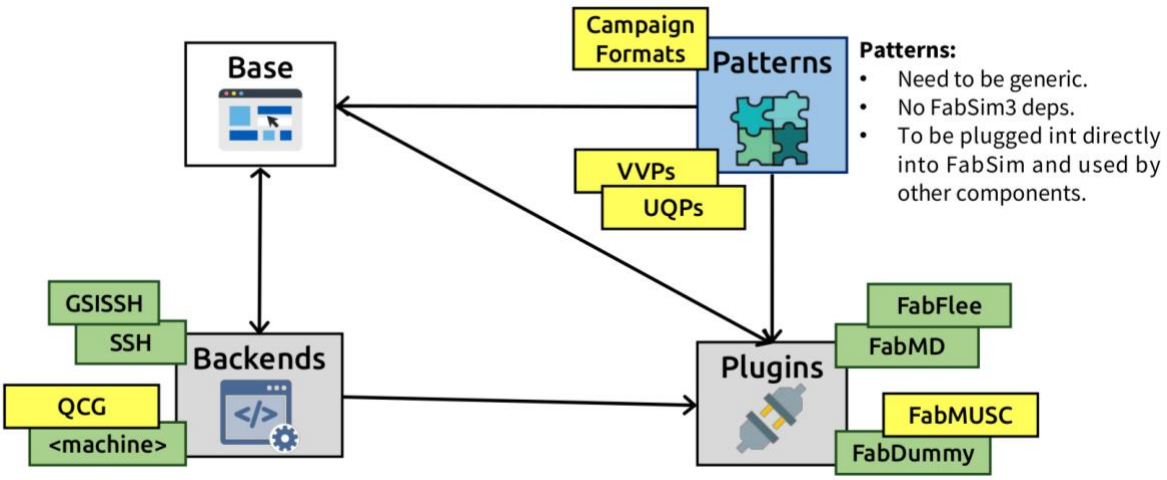

Figure 4: Overview of the FabSim3 architecture.

As a planned scenario, Fabsim3 uses QCG-Client or QCG-SDK to submit a so-called Pilot Job. Such a job can be seen as a container for many subjobs that can be started and managed without individual subjobs having to wait for resources to become available, increasing efficiency. Once the Pilot Job is

submitted, it may service a number of defined VVUQ subtasks (e.g., a set of runs defined by EasyVVUQ). The QCG Pilot Job mechanism provides two interfaces that may be used interchangeably. The first one allows to specify a file with the description of subjobs and execute the scenario in a batch-like mode, conveniently supporting static scenarios. The second interface is offered with the REST API and it can be accessed remotely in a more dynamic way. It will be used to support scenarios where a number of replicas and their complexity dynamically changes at application runtime.

The QCG software (www.qoscosgrid.org) provides advanced capabilities for execution in a unified way of complex jobs on infrastructure consisting of one or many computing clusters. The QCG infrastructure, which is presented in Figure 5, features at the heart the QCG-Broker service, which is responsible for managing execution of the computational experiment, including the multi-criteria selection of resources. In addition, several QCG-Computing services offer unified and remote access to underlying resources. The QCG services can be accessed with numerous user-level tools (Piontek, 2016), of which a few examples are provided in the aforementioned figure.

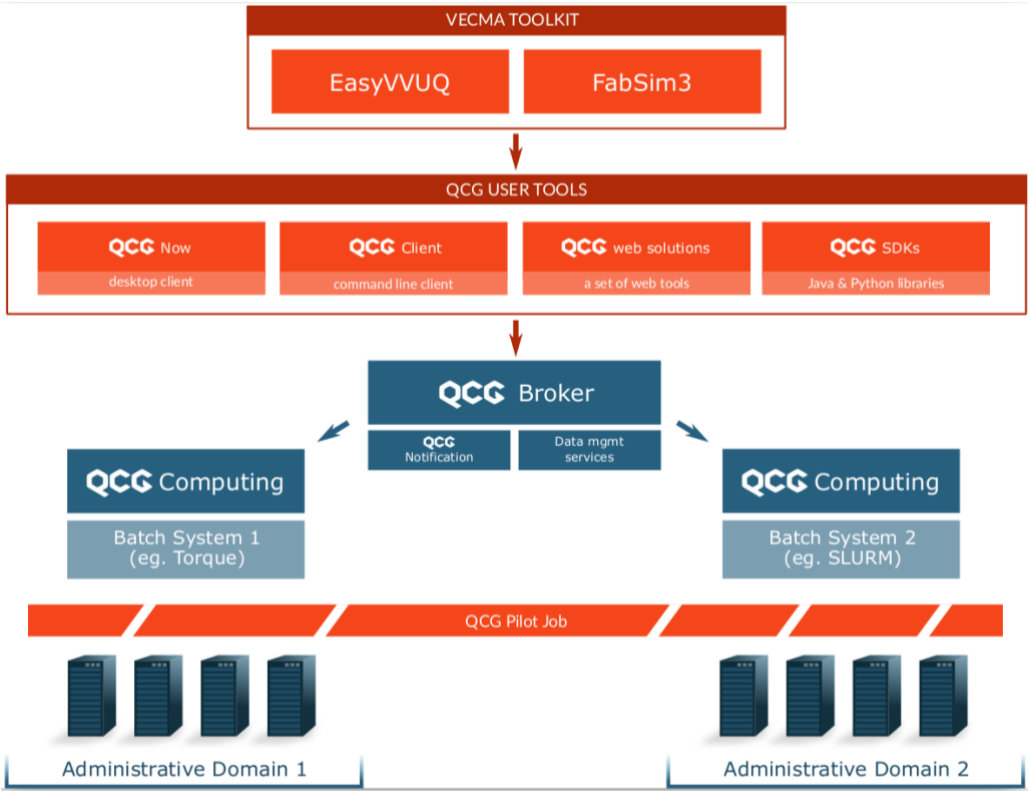

Figure 5: Simplified overview of the QCG infrastructure.

2.3.2 Roadmap and release strategy

As part of the WP3 activities, we have developed a software release plan for VECMAtk, which contains the release schedule. It is intended for the consortium to use and share to remind ourselves for the due date at various stages of the release. In terms of how we perform the release, we have also created a VECMAtk Roadmap to serve as a general guideline, with related activities such as conferences and training planned to disseminate and support the software released throughout the project life period. These documents are not official project deliverables, yet their purpose is highly relevant here. Thus, we highlight the key parts of those documents here:

For VECMAtk we have adopted a release schedule (see Table 1) with two type of releases. Minor releases are tagged every 3 months, advertised within the consortium, and made available to the public without dedicated advertising. These releases incorporate new functionality and components, and are accompanied with a limited amount of additional documentation and examples. Major releases will be made in June 2019, June 2020 and December 2020, advertised and made available publicly. The major releases are accompanied with extensive documentation and examples, and we will hold training events and other dedicated uptake activities in association with these releases. We are currently able to guarantee formal support for VECMAtk only up to June 2021, providing us with 6 months of time to future-proof VECMAtk after the full release in M30. The June 2019 VECMA toolkit release will contain FabSim3, EasyVVUQ, as well as functionalities provided by QCG, and we expect additional components to be introduced in later releases. In between releases, users are able to check out any version of the code for FabSim3 and EasyVVUQ, as we are maintaining an open development environment for both these tools.

Deliverable	Name				Due
D3.1, Public	**Fast-track release of the WP3 VECMA toolkit**				**M12 14-06-2019**
Consortium and alpha users	M3 14-09-2018	M6 14-12-2018	M9 14-03-2019	M11+2weeks 01-06-2019	
D3.3, Public	**Preliminary deep-track release of VECMAtk**				**M24 14-06-2020**
Consortium and alpha users	M15 14-09-2019	M18 14-12-2019	M21 14-3-2020	M23+2weeks 01-06-2020	
D3.4, Public	**Full release of the entire VECMA toolkit**				**M30 14-12-2020**
Consortium and alpha users	M27 14-09-2020	M29+2weeks 01-12-2020			

Table 1: Software release schedule for VECMAtk. The release dates are approximate, and may vary by up to a month to allow the consortium to time releases for optimal impact, and/or to avoid holiday periods.

In the Roadmap, we describe the main aims and activities that we seek to undertake as part of each release. As a baseline we include all the goals stated in the proposal, but the roadmap also provides much more detail on how we seek to establish the toolkit in the wider community through effective dissemination. Most of the Roadmap content is primarily relevant to WP6, but one aspect (concerning the alpha users) is worth elaborating on here.

2.3.2.1 Alpha users

A common concern for many scientific software projects is a lack of uptake and feedback from external users, particularly early in the project. Within VECMA we therefore chose to identify and support a growing number of so-called *alpha users*. Alpha users are potential VECMAtk users within, or outside of, VECMA, who are interested in using the toolkit early on and providing guidance on its development. Alpha users undertake to test the toolkit and provide us with feedback after each three-monthly

release, and to apply parts of the toolkit to their own applications where they consider this relevant. In return, we announce every minor release to them, and give them support in both using the toolkit effectively, and in dedicating effort to incorporate any reasonable feature or documentation request that they may have. Although alpha users test the software, we do not want them to spend extensive time debugging it. Therefore, we generally recommend them to complain whenever certain components are not working, and only retry them at a much later stage when it convenient for them. In the meantime, the development team is expected to have addressed the underlying issue, provided adequate documentation and, where possible, ensured a clarified and rectified situation by securing the relevant features under continuous integration testing. So far, three alpha users have provided us with feedback on the M3 release, and five alpha users on the M6 release.

2.4 Current status of technical development

In the previous section, we provided a high-level overview of the toolkit, and how we intend to accelerate uptake by incorporating alpha users from the get-go. In this section we will describe in more technical detail what we have accomplished in terms of development up to this point.

2.4.1 Brief summary of activities over time

In this subsection we summarize the technical activities that we have specifically undertaken in the months leading up to M9 in the project.

2.4.1.1 Before M1

In preparation of the VECMA kick-off, partners UCL and UBRU invested additional effort to ready a pre-release of FabSim3 already by Month 1. FabSim3 is a successor version of FabSim (http://www.github.com/UCL-CCS/FabSim), which contains a range of updates that we considered absolutely essential in order to get started with the technical activities in VECMA. The updates that we incorporated in this period include:

- Migration from Python 2 to Python 3.
- A more systematic plug-in system, which helps prevent customization of the toolkit from eroding the main code base.
- Better documentation.
- A FabDummy plugin, to provide a few clear and simple testing examples of FabSim3, how to customize it, and how to create your own plugins.
- Support for Eagle, Prometheus, ARCHER and SuperMUC supercomputers, among others, to ensure users are not bottlenecked by lack of HPC resource support.
- Code cleanup, to rid the code of features that were no longer used, or caused major issues with the toolkit as a whole.

The Month 1 release supported three plugins: FabDummy (open) for testing, FabFlee (access on request) for agent-based modelling applications, and FabMD (access on request) for materials molecular dynamics applications.

2.4.1.2 M1 - M3

In this period we further prioritized increasing the robustness of FabSim3, and began work on one of the most essential features for VVUQ procedures, namely the ability to launch multiple jobs in a streamlined manner. It was in this period that we first proposed introducing alpha users, to help reduce the barrier of entry to our toolkit. As a result of these activities, we released an M3 version within the consortium, which added:

- Support for ensemble job execution, allowing sets of jobs, each with a different input file, to be run on remote resources.
- Built-in continuous integration, allowing the code base to be automatically tested on the Travis-CI cloud with every new commit.
- Improved plug-in support, making it easier to install and remove plugins.
- Improved documentation across the board, to make it easier for alpha users to understand how they could use the toolkit.

The M3 version was released within the consortium on 14-09-2018, according to plan.

2.4.1.3 M4 - M6

We received our first alpha user feedback in M4, which was abundantly clear: although the alpha users could clearly identify the potential of the toolkit, a range of bugs, documentation issues, and incomplete feature-sets prevented them from using FabSim3 effectively. During this period we therefore incorporated a range of features to help address these issues. Compared to the previous release, the M6 release contained the following improvements in FabSim3:

- Improved documentation, more examples.
- More intuitive ensemble execution support (e.g. sweep across directories).
- Ability to set plugin-specific remote modules.
- Ability to cancel jobs.
- Ability to check if jobs have completed.
- Fixes in FabDummy to make testing easier.

One other requirement stood out in particular. Although the automation provided by FabSim3 covered some of the user needs, most notably the toolkit was lacking the means to easily construct a parameter space, and create a batch of job definitions to explore this parameter space in a given way. This is essential in any application that needs to be tested e.g. for parameter sensitivity, or for how different (uncertain) values in its input propagate to uncertain values in its output.

To address this concern, we developed a new component, EasyVVUQ, and added this to the M6 release of VECMAtk. EasyVVUQ is open development, and can be found here at any time: http://www.github.com/UCL-CCS/EasyVVUQ.

The M6 release of EasyVVUQ (v0.1) contains a basic architecture for UQ workflows, including elements for:

- Reading input defining the parameter space of interest.
- A *Campaign* object to record runs required for analysis.
- Samplers to create runs required for a given analysis workflow.
- Encoders to help create simulation inputs.
- Decoders to help read and interpret simulation outputs.
- Collation of output from multiple runs into a pandas dataframe for analysis.
- Simple statistical analysis.

The M6 version of VECMAtk was released to the consortium on 14-12-2018, according to plan.

2.4.1.4 M7 - M9

The M6 release was well received overall. Not only did we receive feedback from 5 alpha users, but users external to the consortium also attempted to obtain the toolkit, even though the release was internal. After a brief discussion, we decided to modify our release approach to cater for this demand, and as of 14-01-2019 not only EasyVVUQ, but also FabSim3 became open development, allowing anyone on the internet to clone the code, raise issues, and access releases that were previously labelled as internal.

In the M7 - M9 period we shifted priorities in several areas:

A. More effort was dedicated to writing instead of development, to allow for the completion of D3.1, D4.1, D5.1, and the VECMAtk paper which has been submitted to the International Conference on Computational Science, and which we would like to accompany the M12 release.
B. Due to particularly high demand for EasyVVUQ, and lack of development effort there, we shifted effort partially away from FabSim3 towards bolstering EasyVVUQ.
C. For FabSim3, the remaining effort available was mostly dedicated to adding support for QCG, which we seek to include in the M9 release, and working towards incorporating the first UQP.

For the M9 release we currently expect the following features to be present:

- FabSim3: Support for containerization (with FabSim3 deployable as a Docker Image).
- FabSim3: Support for job submission using QCG Broker.
- EasyVVUQ: Streamlined implementation supporting a much wider range of applications.
- QCG-Client: Basic scripts that allow users to install the latest version of QCG-Client as convenient.

In addition, we aim to support a wider range of applications, and to have implemented a first version of a UQP (Non-Intrusive Monte Carlo). The M9 release is planned for March, and will be public (though not as widely advertised as the M12 release will be).

2.4.2 Summary of meetings during M0 - M9

During the first nine months of the project, we organized a range of face-to-face technical meetings that helped us accelerate our progress. These include:

- VECMAtk meeting (17 October 2018), where we received feedback from alpha users in person, and worked out the design for ensemble execution in practise, as well as the Software release plan and the Roadmap.
- The Dev team leaders meeting via Skype (25 October 2018). Here we reached agreement on when to include QCG, EasyVVUQ and other tools in VECMAtk. We agreed at that time to include EasyVVUQ in M6, QCG in M9, and other components at a later stage.
- The VECMA project wide meeting, Garching, Munich (4 December 2018). Here we updated the rest of the consortium about our progress, and put forward our proposition to rename the "P" in UQPs and VVPs from "Primitive" to "Pattern". This is because we realized during the development of EasyVVUQ that each UQP and VVP consists of subunits, which we labelled "elements". As such, the definition of the "P" as primitive became technically incorrect, as it lost its atomic nature. We also identified here the pressing need to add support for Pilot Job execution, which enables the scheduling and execution of many small jobs within a large job allocation slot on a single resource.
- The VECMA meeting on Verification and Validation Primitives (11-12 December 2018). Here we formulated our schedule in regard to the implementation of VVPs (to be started M12) and formed a new development team, focusing exclusively on Pilot Job implementation in VECMAtk.

2.4.3 Overview of testing integration and usability activities

Testing is an important aspect of creating high-quality software and deployment perspectives. To ensure that any commit made to the repository on Github has not broken any prior functionality, we require Continuous Integration (CI), a process of running tests and reporting the results every time you push a new commit. In this project, we use Travis CI (www.travis-ci.org) for continuous integration. Travis CI is an open-source hosted, distributed, and continuous integration service used to build and test projects hosted at GitHub. We present a high-level overview of the process in Figure 6. Travis CI supports the development process by automatically building and testing code changes, providing immediate feedback on the success of the change.

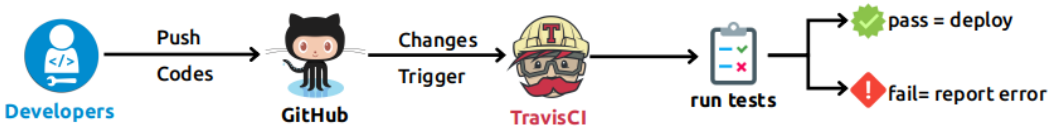

Figure 6: VECMAtk development process for pushing and testing changes.

To effectively support user mobility to run the toolkit regardless of the infrastructure or additional requirements, we containerized our toolkit using Docker and Singularity. Containerization is an operating system level virtualization method used to deploy and run distributed applications without launching an entire virtual machine (VM) for each application. Containers allows us to create a virtual machine (VM) in very lightweight fashion compared to the traditional virtualization approach, as they share the operating system kernel, and they also share resources such as storage and networking. Additionally, application containers may be migrated to other computing environment, such as computing system, clouds, or other environments without requiring code changes.

Each container includes the runtime components -- such as files, environment variables and libraries -- necessary to run the desired software, and run on a single control host, accessing a single kernel. In VECMA, we initially use Docker to containerise our toolkit, as it is easy to use, and also plan to containerise using Singularity, which is increasingly widely supported on HPC resources. We describe both these efforts below.

Docker is a powerful and lightweight container management tool, which provides an easy way to package an application in a container. A typical Docker workflow to create images, pull images, publish images, and run containers from a `Dockerfile` is shown in Figure 7A. A `Dockerfile` is a script that contains collections of commands and instructions that will be automatically executed in sequence in the docker environment for building a new docker image. For example, your container could be a based on Ubuntu OS running a version of python with pre-installed library required by your application. This container can be started and stopped in the same was as one would start and stop a VM. Finally, we can push our container image to a Docker registry, i.e, Docker Hub, for the world to use.

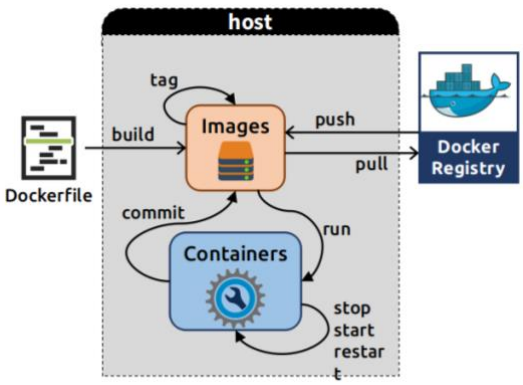

Figure 7A. A typical Docker workflow

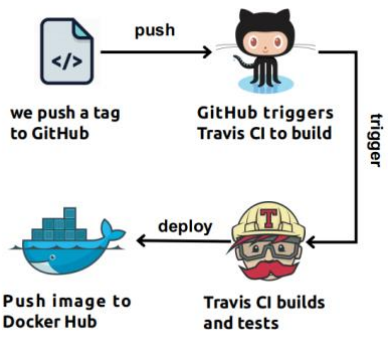

Figure 7B. GitHub + TravisCI + Docker

Currently, we set up the FabSim github repository with a `Dockerfile` and Travis file, as shown in Figure 7B. After each successful test executed by Travis, based on the `Dockerfile`, a docker image is configured and built, and pushed into Docker Hub. For this work, A Docker image is provided as well through the Docker Hub[1]. And, the docker bundle for easy deployment is available in Docker Hub via :`docker pull vecmafabsim3/fabsimdocker`.

Singularity. Although Docker is one of most popular container technologies for software reproducibility, it has low adoption in the HPC world since it requires users with root access to run Docker and execute a containerised applications. To support high performance computing use cases, where users should only have access to their own data, Singularity[2] can be used as a solution for container system in HPC environment. Singularity containers differ from Docker containers in several important ways, including the handling of namespaces, user privileges, and the images themselves. As part of the ongoing work, we will provide a singularity container alongside a docker image for VECMAtk, and at time of writing we are testing Singularity containers for FabSim3.

2.5 Overview of upcoming releases

VECMAtk is scheduled to be released in months 12, 24 and 30 in the project, and we aim to have a high impact with each release. By the final deep-track release version in M30, we will implement all UQPs and VVPs from WP2 as working building blocks within VECMAtk, and have working integrations with coupling tools such as MUSCLE, facilities for Pilot Job submissions and to the WP5 middleware. In the spirit of evolutionary prototyping, we will use the M24 VVUQ release (preliminary deep-track) to gather community feedback and calibrate our documentation efforts, and unveil the main release on M30. After M30 we will focus on maintaining the deep track toolkit, enabling the incorporation of performance optimizations, stability improvements, improved documentation, and new UQPs/VVPs. In the rest of this section we provide a brief overview of what we expect to be present in the three major VECMAtk releases.

2.5.1 Release M12: the fast-track release of the VECMA toolkit

This release will contain the fast-track functionalities, including unit and functional tests, as well as documentation and tutorials. It will contain EasyVVUQ, FabSim3 and several QCG tools. For this release, we have the following objectives:

[1] https://hub.docker.com/r/vecmafabsim3/fabsimdocker
[2] https://singularity.lbl.gov

- Perform the release concurrent with the ICCS 2019 conference (and the MMS workshop) in Portugal in mid-June 2019.
- Support the first non-intrusive UQPs in the toolkit, such as stochastic collocation and Monte-Carlo based sampling.
- A tentative approach for constructing VVUQ procedures (part of the "VVUQ Procedure Builder" milestone).
- Support for all machines in the VECMA infrastructure, and at least rudimentary support for all active applications in the project.
- Integration: EasyVVUQ with QCG-pilot, FabSim3 with QCG-Client and QCG-Broker.
- Provide sufficient documentation to allow for systematic uptake by users, as well as Docket and Singularity images.
- Support for 5 different HPC resources.

2.5.2 Release M24: the preliminary deep-track release of the VECMA toolkit

This is a preliminary software release of VECMAtk which contains deep-track functionalities, including unit and functional tests. For this release we have the following objectives:

- Provide VECMA Toolkit documentation: a "living" deliverable providing documentation and tutorials on this release of VECMAtk, and ensuing ones.
- Implement all UQPs and VVPs that have been finalised by WP2, and have examples for each of them involving at least one application.
- Deliver a final release of EasyVVUQ (subsequent development effort in this area will be dedicated towards more deep-track mechanisms). This constitutes another major step in delivering the VVUQ procedure builder as detailed in D5.1 and the proposal, and a preliminary version will be included in this release.
- Support pilot job mechanisms, enabling more efficient execution of the UQPs and VVPs on HPC resources.
- Support for all applications in the project.
- Incorporate the first optimizations to explicitly make the toolkit more scalable in terms of number of jobs and execution efficiency.
- Achieve uptake with users outside of VECMA, and a wide and effective outreach effort.

2.5.3 Release M30: The full release of the entire VECMA toolkit

The M30 software release will contain deep-track functionalities, including unit and functional tests. Community feedback on the M24 release will be used to shape this main

release. Because of this we cannot yet define the full list of objectives. However, objectives that we expect to include are:

- Comprehensive / final versions of all components involved.
- Support for intrusive / semi-intrusive UQPs and VVPs.
- Incorporate all necessary optimizations to make the execution of VVUQ procedures for multiscale applications as efficient as possibly on current infrastructures.
- Incorporate optimizations to enable efficient and effective use of VECMAtk for future (or by then current) exascale infrastructures.

3 Conclusions

We presented the development approach adopted in VECMA, an overview of the current status of the VECMA toolkit software, as well as our objectives for the M12,M24, and M30 toolkit release. In terms of technical progress, we argue that our activities are on track in regards to the milestones identified in the proposal. By dedicating effort to continuous integration, containerization, and cross-component integration already in these first nine months, we have been able to expose new functionalities for existing application developers and users earlier than anticipated. As a result, we generated more user interest in the toolkit than we anticipated in advance. Indeed, in terms of user uptake we are slightly ahead of schedule, with some external users (e.g., researchers in Petros Koumoutsakos' group at ETH Zürich) already having tried out our software. We believe that overachievement in uptake is always a good thing, and have, as a first step, switched to open development to further foster the nascent VECMAtk community.

The distributed development of a VVUQ toolkit for multiscale HPC applications is a challenging endeavour indeed, and we find ourselves continually switching priorities and rearranging people in order to forge solid progress on the most important fronts, at the most important moments. Delivering the UQP/VVP implementations, systematic continuous integration, ease of installation, support for applications in any domain, a growing enthusiastic and responsive user base, and software that scales in several aspects (e.g., # of models, # of nodes, # of jobs, # of TBs of data); though not all of it is written as milestones in the proposal, all of it *is* quintessential to make this toolkit a success.

In the first 6 months we focused especially on a robust and accessible implementation of two tools (FabSim3 and EasyVVUQ) and getting the first users on board. Currently, we are focused on including even more applications and incorporating the added value provided by the QCG

software tools and middleware. Between now and M36 our priorities are bound to shift many more times, and it is due to the very nature of evolutionary prototyping that many requirements and approaches are emergent rather than hard-coded in advance. But given the substantial progress we have made so far, and the interest and uptake which is currently increasing beyond our expectations, we believe we have chosen the right route. At least for the time being, of course.

4 References

(Delgado-Buscalioni, 2003) Delgado-Buscalioni, R., and Coveney, P.V.: USHER: an algorithm for particle insertion in dense fluids. *The Journal of chemical physics* **119**.2 (2003): 978-987.
(Groen, 2016) Groen, D., Bhati, A.P., Suter, J., Hetherington, J., Zasada, S.J., Coveney, P.V.: Fabsim: Facilitating computational research through automation on large-scale and distributed e-infrastructures. *Computer Physics Communications* **207** (2016) 375–385.
(Groen, 2018) Groen, D., Richardson, R.A., Wright, D.W., Jancauskas, V., Sinclair, R., Karlshoefer, P., Vassaux, M., Arabnejad, H., Piontek, T., Kopta, P., Bosak, B., Lakhlili, J., Hoenen, O., Suleimenova, D., Edeling, W., Crommelin, D., Nishikova, A. Coveney, P.V.: Introducing VECMAtk - verification, validation and uncertainty quantification for multiscale simulations *(submitted)*.
(Piontek, 2016) Piontek, T., Bosak, B., Ciznicki, M., Grabowski, P., Kopta, P., Kulczewski, M., Szejnfeld, D., Kurowski, K.: Development of science gateways using QCG — lessons learned from the deployment on large scale distributed and HPC infrastructures. *Journal of Grid Computing*, **14** (4) (Dec 2016) 559–573.
(Sobol, 1998) Sobol, I.: On quasi-Monte Carlo integrations. *Mathematics and Computers in Simulation* **47** (2) (1998) 103 – 112.