D5.1: Architecture of the VECMA system

Due Date	14 March 2019
Delivery	12 March 2019
Submission of updated version	N/A
Lead Partner	PSNC
Dissemination Level	Public
Status	Final
Approved	Executive Board
Version	1.0

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
14.01.2019 v0.1	Bartosz Bosak	The document top-level structure - TOC, Initial version of architecture diagram
28.01.2019 v0.2	Bartosz Bosak	Architecture diagram improved and divided into 3 layers
29.01.2019 v0.3	Bartosz Bosak	New image with generalized view on the architecture
1.02.2019 v0.4	Tomasz Piontek	Corrections and extensions of the text
11.02.2019 v0.5	Bartosz Bosak	Accepted input from partners
18.02.2019 v0.6	David Wright	Language corrections
19.02.2019 v0.7	Tomasz Piontek	Preparation of the document for the internal review
01.03.2019 v0.8	Tomasz Piontek, Bartosz Bosak	Addressed internal reviewers comments
08.03.2019 v0.9	Peter Coveney	Final remarks and comments
13.03.2019 v1.0	Tomasz Piontek, Bartosz Bosak	Final corrections

CONTRIBUTORS

- Tomasz Piontek, PSNC – Main Author
- Bartosz Bosak, PSNC - Author
- Piotr Kopta, PSNC – Author
- Krzysztof Kurowski, PSNC - Author
- Paweł Spychała, PSNC - Author
- Derek Groen, Brunel University - Author
- Onnie Luk, IPP - Author
- Erwan Raffin, Atos-Bull - Author
- David Wright, UCL - Author
- Robin Richardson, UCL – Author

Disclaimer

This document's contents are not intended to replace consultation of any applicable legal sources or the necessary advice of a legal expert, where appropriate. All information in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user, therefore, uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission has no liability in respect of this document, which is merely representing the authors' view.

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

DEFINITION AND ACRONYMS

Acronym	Definition
API	Application Programming Interface
AMUSE	Astrophysical Multipurpose Software Environment
HPC	High Performance Computing
HPMCP	High Performance Multiscale Computing Patterns
MMSF	Multiscale Modelling and Simulation Framework
MUSCLE	MUltiScale Coupling Library and Environment
OMUSE	Oceanographic MUltipurpose Software Environment
PRACE	Partnership for Advanced Computing in Europe
QCG	Quality in Cloud and Grids
SDK	Software Development Kit
VVUQ	Verification Validation and Uncertainty Quantification

1 Executive summary

This deliverable presents the architecture of the infrastructure and tools being developed within VECMA. The architecture consists of the three layers which, in line with the design proposed in the VECMA DoA, correspond to different Work Packages within the project: **Applications** (WP4 - Applications), **VECMA Toolkit** (WP3 - Software implementation) and **Execution** (WP5 - Infrastructure). The aim of this deliverable is to provide an overview of how VECMA will meet the requirements of the diverse set of application scenarios for multiscale Verification, Validation and Uncertainty Qualification (VVUQ) by the mitigation of the current infrastructure limitations.

The design of our architecture was informed by a requirements gathering exercise that was initiated through a questionnaire used to understand the needs of the exemplar applications developed within WP4. This document, based on the responses received, and in accordance with the generally accepted methodology of software design and development, provides a reference point responding to the VECMA objectives with the aim of satisfying the needs of the project partners.

Whenever both possible and beneficial, VECMA builds upon the achievements of previous projects and already available open-source software. Thus, the architecture, along with proposed new components related specifically to multiscale VVUQ requirements, includes a range of existing software that will be reused with or without modifications. This document provides a clear distinction between new VECMA components and those which have been or are being developed external to the project. Moreover, for each of the selected components, a short summary describing its features and role in VECMA is available as well as expected developments required to use a particular component in defined VECMA application scenarios are presented.

In order to offer application groups the essential functionality in the shortest possible time, the proposed VECMA architecture distinguishes between fast track and deep track components and capabilities. The components assigned to the fast track will be developed, integrated and released by M12 of the project and will form a usable environment for the basic experiments and fundamental VVUQ analysis requested by the various application teams. This release will also form the backbone for further releases of the VECMA tools and services. The deep-track components will be deployed subsequently based on support for more sophisticated scenarios.

This deliverable presents the concepts and starting points for all subsequent technical developments in VECMA, and therefore constitutes an important early milestone in the project. Nevertheless, the proposed architecture may be subject to changes (adaptations and extensions) during the course of the project as a result of both possible conceptual changes and the emergence of additional requirements.

2 VECMA architecture

The principal objective of VECMA is to enable efficient and verified multiscale calculations on high-end (approaching exascale) computing infrastructure. This deliverable summarises the work done in WP5 to define a comprehensive VECMA system architecture. The outline presented here provides a road map for all future developments within the project, in particular for the software provisioners from WP3 and WP5. Our goal is to define how we will build a fully functional system from a set of diverse software components (both pre-existing and newly developed) for Validation, Verification and Uncertainty Quantification (VVUQ). To that end this section sets out how the system will be designed and implemented on a technical level in order to execute the VECMA's grand challenge, exemplar applications on the project's HPC resources.

2.1 Overview

The design of a fully functional and automated system for the execution of multiscale applications on current petascale and emerging exascale resources in such a way that simulation results are obtained with associated uncertainties is not a trivial task. In order to realise this objective two essential challenges must be overcome by VECMA (and by WP5 in particular):

1. To execute efficiently coupled applications consisting of a potentially dynamic number of sub-tasks on thousands or even millions of cores;
2. To develop and integrate scalable VVUQ procedures for automated execution of those multiscale applications on e-Infrastructure and analysis of produced results.

Fortunately, to address these challenges we are able to benefit from close cooperation between application developers, system architects and infrastructure providers in VECMA, as well as the expertise gained by some VECMA partners in a number of previous EU funded projects, particularly MAPPER[1] and ComPat[2].

Thanks to the cooperation between different work packages realised in VECMA, and especially on the basis of the results of a survey in which application teams described their use cases and expectations, it was possible to collect precise requirements for the developed system. The part of the questionnaire presented to the application developers which directly relates to issues affecting system architecture is attached in Annex 2. Reference to the full questionnaire is given in Deliverable D4.1 [1]. Furthermore,

[1] http://mapper-project.eu/

[2] http://compat-project.eu/

the aforementioned expertise of the VECMA partners allowed us to ground the architecture in a number of well tested design principles and integration schemes, as well as providing already mature software components. The major outcomes of previous projects which will facilitate our work in VECMA are the Multiscale Modelling and Simulation Framework (MMSF) [2] developed in MAPPER, and the High Performance Multiscale Computing Patterns (HPMCP) [3] developed in ComPat. With incorporation of these tools into the VECMA software stack, the time and effort needed to provide anticipated solutions will be significantly reduced, at the same time increasing the probability of success of the whole project.

A high level description of the architecture, which brings together the work of WP3, WP4 and WP5, is shown in Figure 1. The overall system, can be divided into three partially overlapping layers: Applications, VECMA Toolkit and Execution.

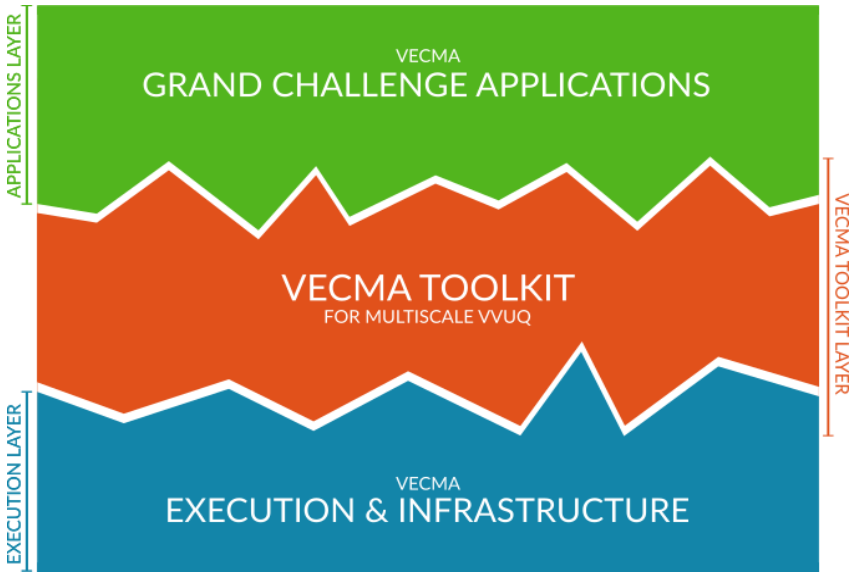


Figure 1 The three main "layers" of the VECMA system

The VECMA Toolkit layer is the main focus of our research and is anticipated to be the main outcome of the project. In this layer, Verification, Validation and Uncertainty Quantification methods, designed by WP2 (Algorithms & Formalisms), are introduced alongside tools which facilitate their use in multiscale models. The majority of the effort required to realize the toolkit will occur within WP3 (Software Implementation). The main design principle followed in this layer is to ensure that all components developed are as reusable as possible and do not depend on any particular method of execution. Whilst this may imply that the toolkit is independent of the Applications and Execution layers, conducted by WP4 (Applications) and WP5 (Infrastructure) respectively, in practice the features of the toolkit are derived from the requirements of the exemplar applications and the infrastructure tools will make the efficient execution of the developed methods possible. In other words, the requirements of the Infrastructure layer depend on the methods developed in the VECMA Toolkit layer

and these methods in turn depend on the applications they support. Thus the VVUQ paradigms are considered across all the technological layers.

A full, albeit simplified, picture of the developed architecture of the VECMA system is shown in Figure 2.

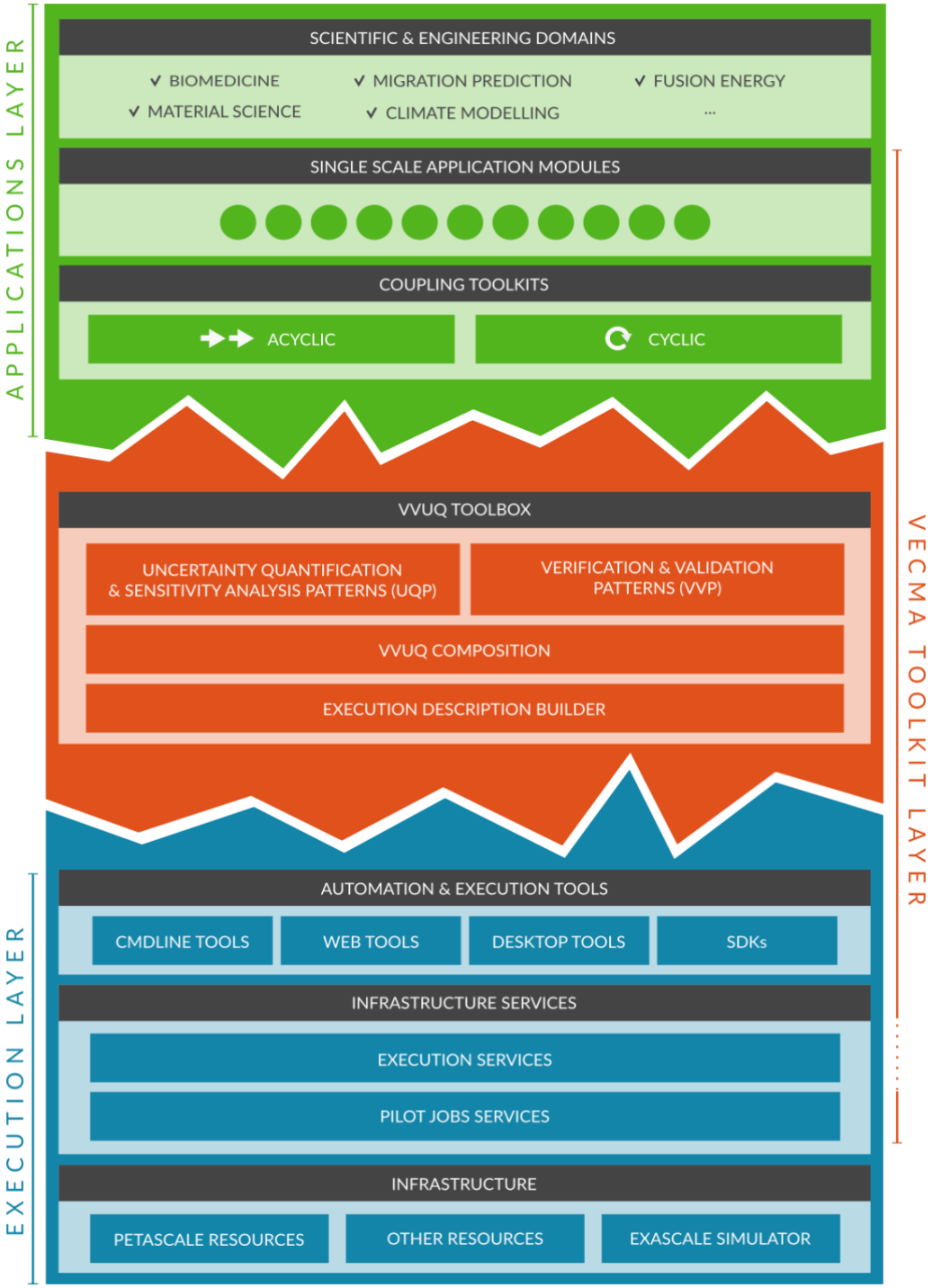


Figure 2 Overview of the architecture of the VECMA system.

The individual layers of the architectural stack will be discussed in detail and presented separately in the following sections.

In the remainder of the document, within each layer, the following colour scheme will be used to indicate the development status of different components of the VECMA system. Blue boxes are used to mark existing elements or ones external to the project that will be reused in VECMA out-of-the-box or after minor adaptations and extensions. Orange and red boxes are reserved for components whose development has been initiated within VECMA. The orange boxes represent fast track components, i.e. the core components that are being developed earlier in the project and will be included in the first releases of the VECMA system scheduled to be in M12. Red elements are the deep track components - including the more advanced (e.g. semi-intrusive and intrusive VVUQ) or auxiliary VECMA components - scheduled to be released in the later phases of the project, after M12. Moreover, this document makes use of different shapes for marking features of particular components; boxes with sharp corners are used to indicate generalised conceptions or abstract procedures while boxes with rounded corners represent implemented software components.

Detailed information on the development of individual components is provided as Annex 1.

2.2 Applications Layer

The VECMA project is application-driven; therefore, the real scientific requirements are of prime importance in the development of the VECMA system. For this reason, the top-level position in our architecture is given to the Application layer, which is graphically depicted in Figure 3. The key point is that the VECMA system is designed to support multiscale applications (typically made up of multiple single scale models) from a wide range of scientific, engineering or even humanities domains, with common components that provide domain agnostic functionality such as cyclic or acyclic model coupling.

2.2.1 Scientific & engineering domains and their applications

The exemplar (multiscale, component based) applications within the VECMA project can be categorized into five major scientific and engineering domains: biomedicine, material science, migration prediction, climate modelling, and fusion energy.

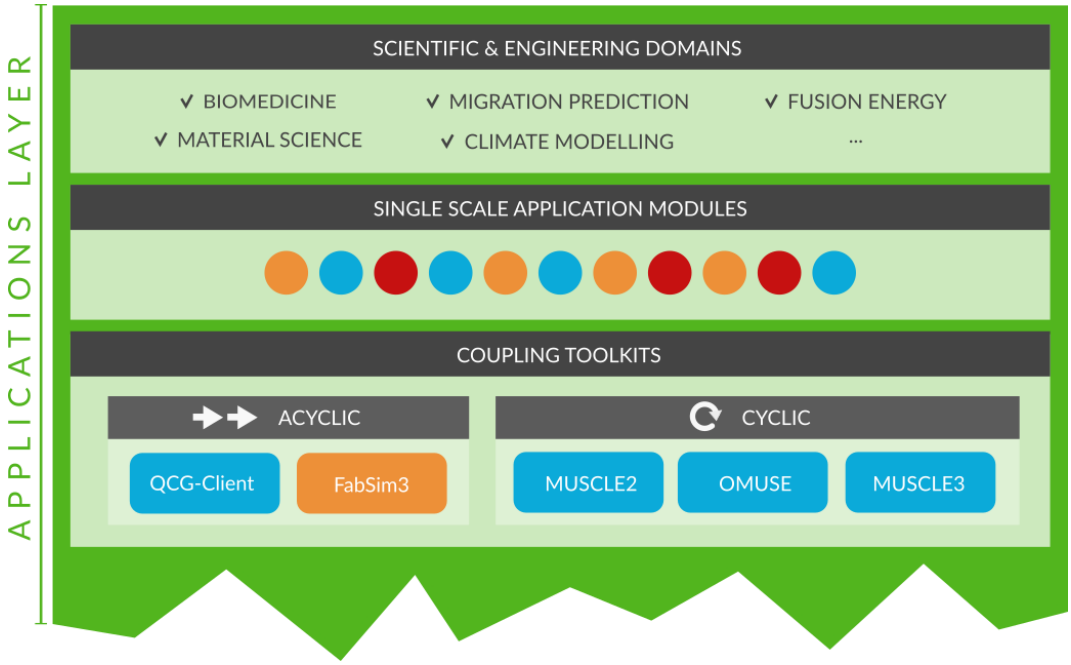


Figure 3 Architecture of the Applications layer

There are four applications within the biomedicine domain. HemeLB is a high performance, parallel lattice-Boltzmann code that simulates fluid flow in complex geometries such as cerebral blood flow. The In-Stent Restenosis model (ISR3D) is a fully-coupled 3D multiscale model of in-stent restenosis, with blood flow simulations coupled to smooth muscle cell proliferation. HemoCell model is a versatile 3D cell based blood flow model, which can be coupled to macroscale models of continuous bloodflow. The Binding Affinity Calculator (BAC) is a molecular dynamics simulation and binding free energy workflow application for estimation of the binding strength of small molecule ligands to their target proteins using ensemble molecular dynamics simulations. The materials science exemplar application is a concurrent multiscale model that simulates a material's mechanical behaviour using the Finite Element Method and Molecular Dynamics. A multiscale migration prediction (MMP) model within the mass migration domain forecasts movements of refugees when a conflict erupts in order to guide decision-making processes such as resource distributions. Under the climate domain, a multiscale application (CRAtmos) takes the global atmosphere model (OpenIFS) and couples it to the local cloud-resolving high resolution models with limited spatial domains (LES). The fusion application aims to study the effects of micro-scale turbulence (typically described through fluid or kinetic models) to the overall macro-scale transport of a fusion device to predict more accurately the temperature profiles in the plasma core.

For further information regarding the applications mentioned here, please refer to Deliverable 4.1 [1]

2.2.2 Coupling toolkits

Many of the coupling functionalities in VECMA are being evolved from those developed within the preceding MAPPER and ComPat projects. We distinguish two different cases: one where the sequence of single scale model executions does not contain a cyclical dependency (acyclic), and one where it does (cyclic). Two first coupling toolkits, namely QCG-Client provided as part of QCG[3] and FabSim3[4], are optimized for acyclic dependencies, and can easily capture complex multiscale workflows of that category. Although they can also be used for cyclic dependencies, the performance obtained is likely to be very limited, unless one of the cyclic coupling toolkits is used. The cyclic toolkits incorporated to VECMA are MUSCLE (versions 2 and 3) and OMUSE. MUSCLE2 [4] supports a wide range of coupling mechanisms on platforms ranging from desktops to supercomputers and adheres to the formalisms defined in the Multiscale Modelling and Simulation Framework [2]. MUSCLE3, currently in development in the e-MUSC project[5], builds upon this work, and extends it to support semi-intrusive UQ, by adding (sub)model ensembles, multiple parallel parameter sets, dynamic coupling scenarios and a more flexible interface. OMUSE (Oceanographic MUltipurpose Software Environment) [5] is an extension of the AMUSE (Astrophysical MUltipurpose Software Environment) framework [6] employed in the ComPat project, and provides a wide range of coupling facilities, such as model coupling, unit conversion, and API abstraction layers for different single scale model type. The cyclic coupling toolkits are also able to support acyclic coupling scenarios to varying extents.

2.3 VECMA Toolkit Layer

The VECMA Toolkit layer, depicted in Figure 4, is a set of software components designed to support Verification, Validation and Uncertainty Qualification of multiscale applications. The central place in this layer is occupied by VVUQ Toolbox, which has a twofold role: to apply Uncertainty Quantification and Sensitivity Analysis Patterns (UQP) and Verification & Validation Patterns (VVP) to the multiscale applications, and to prepare the respective execution configurations of those VVUQ-enhanced applications that will be understandable by the VECMA Execution services. The UQP and VVP will be described in detail in the forthcoming deliverables D2.1 [7] and D2.2 [8]. Since a typical multiscale application is a collection of coupled modules, each of which can be analysed in some level of isolation,

[3] http://www.qoscosgrid.org

[4] http://www.github.com/djgroen/FabSim3

[5] https://www.esciencecenter.nl/project/e-musc

the idea behind VECMA is not to treat the application exclusively as a black-box (non-intrusive VVUQ), but also to make use of the coupling characteristics (semi-intrusive VVUQ) or even alter the application by replacing some submodels by surrogates (intrusive VVUQ) to optimize the overall efficiency of VVUQ analysis.

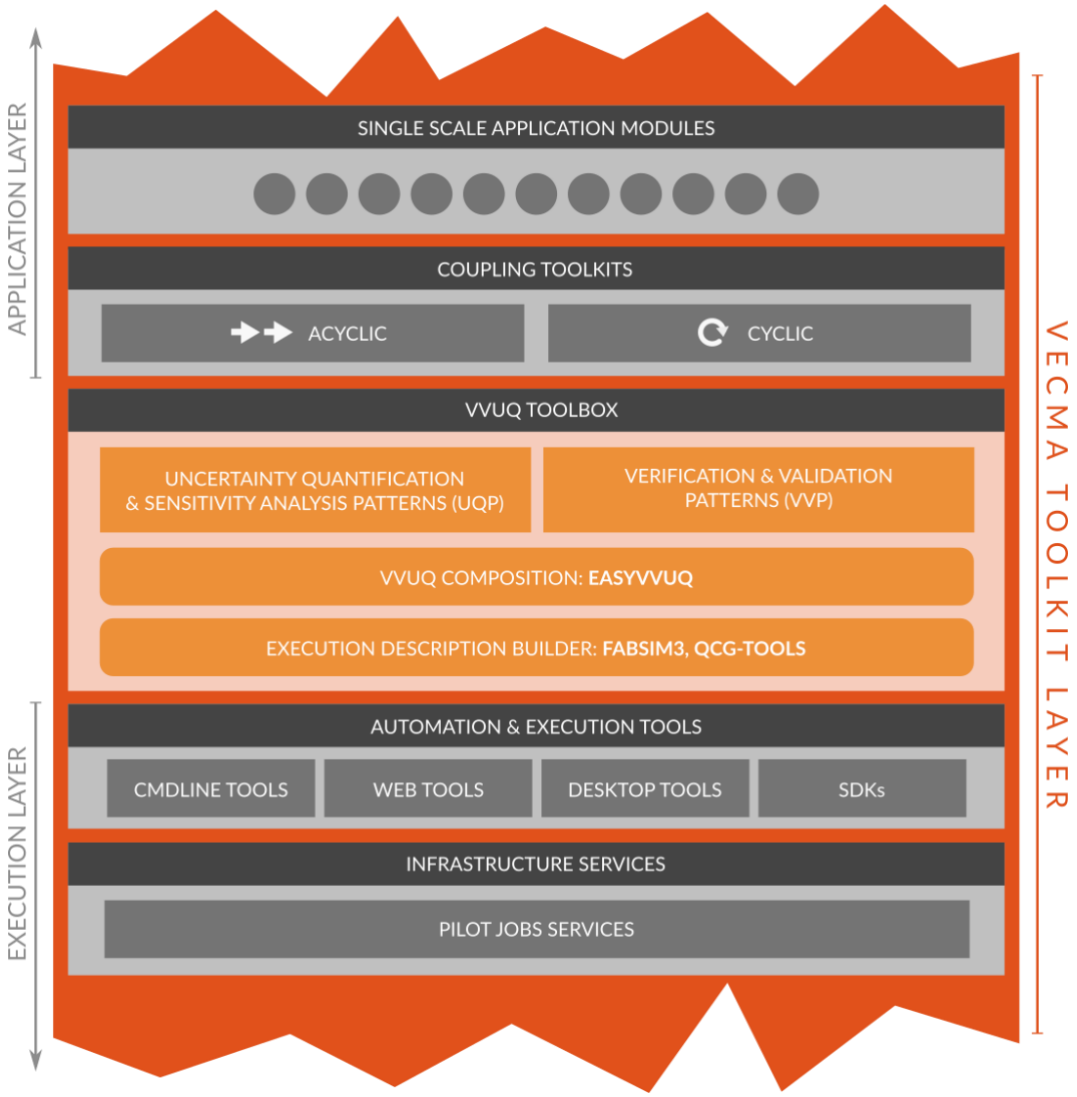


Figure 4. Architecture of the VECMA Toolkit layer

The VVUQ Toolbox is designed to allow a flexible implementation of VVUQ workflows based on the analysis requirements of the scientists developing the applications. In order to simplify the usage of the workflows, where appropriate, the tools will be adjusted to seamlessly integrate with coupling toolkits and the specific input and output data used by applications. Lower down in the architecture diagram, the computing requirements of applications and their associated VVUQ calculations put heavy demands on the Execution layer to ensure calculations are secure and efficient. Here the VVUQ Toolbox also has to cooperate closely with a range of tools in the VECMA Execution layer. For example, the ability to dynamically execute replicas is essential for performing many sophisticated VVUQ

procedures, and this capability is planned to be introduced through the novel Pilot Job mechanisms (for example QCG-PilotJob Manager).

2.3.1 UQP and VVP

Uncertainty Quantification and Sensitivity Analysis Patterns (UQPs), and Verification and Validation Patterns (VVPs), are application-independent and platform-independent definitions of generic UQ and VV routines. They constitute building blocks from which a fully fledged VVUQ procedure (or workflow) can be constructed. More details on the design of patterns will be presented in the WP2 deliverables (for instance in D2.1 [7] scheduled for M12 and D2.2 [8] scheduled for M24), and more details on their implementation will be provided in the WP3 deliverables (for instance in D3.1 [9] scheduled for M9).

2.3.2 VVUQ Composition

The VVUQ composition module serves to allow users to combine various patterns and procedures that can be defined throughout the VECMA toolkit. In the original proposal two separate components were suggested, handling UQ composition and VVUQ procedure building respectively, however as the project has developed it has become clear that the functionality is best considered as a representing a single set of requirements. We have thus combined both into a single VVQU Composition module. The aim of this component is to provide the user with functions that allow them to develop custom VVUQ workflows for their application. The key to this is to abstract the sampling and analysis (required for VVUQ analysis) from the specifics of the input and output formats required by different codes. Application developers will be able to tailor existing workflows when the pattern employed is similar to those created in advance for other applications. Within VECMA, we acknowledge that the software introducing VVUQ procedures into multiscale applications needs to be lightweight and flexible, as these composite procedures will differ heavily per application. It should offer the command-line interface and expose an API which can be used for the development of UI-based composition tools (desktop or web based solutions developed in scope of WP5). At the time of writing, EasyVVUQ (see D3.1 [9] for reference) is our primary tool for supporting this functionality at least to a limited extent, but additional tools are likely to emerge later in the project, in particular to support very complex or dynamic VVUQ procedures.

2.3.3 Execution Description Builder

The Execution Description Builder may be seen as an abstract component whose role is to combine a formalised application specification (including VVUQ procedures and requirements) with application input files and translate it into a suitable form for the execution layer. Since the process of combining

have various aspects it may be realised differently depending on a given use-case, i.e. specificity of the application, the data it produces and consumes, as well as the linked VVUQ procedures and formalisms. It is expected that this component will be realized first as a separate command-line tool, and later, most likely, as a part of integrated command-line and/or graphical solution providing VVUQ composition module functionality. At the time of writing, we have positioned FabSim3 and QCG tools to provide this functionality as a baseline, but it is conceivable that alternatives may emerge as the project progresses.

2.4 Execution Layer

The embedding of VVUQ procedures into complex multiscale simulations has to be supported by services and tools enabling the whole process from efficient execution of calculations on computing infrastructure to data management and security assurance. However, as it was already recognized in the previous projects [2] that advanced multiscale applications require specific capabilities from supporting middleware and often cannot be simply run with commonly used software enabling only basic access to computing clusters. The specific needs of multiscale computing, to name just a few, are: requirement for simultaneous execution of many large application modules, requirement for heterogeneous resources for running parts of an application and assurance of good communication efficiency between distributed processes. In addition, irrespective of the mechanisms necessary to perform the multiscale application itself, the role of middleware services is to ensure the required quality of service by making an optimal allocation of tasks to computational resources in a way that optimizes the often conflicting criteria set by the user and the resource provider: for example, to achieve results in the shortest time and with the lowest energy consumption. In VECMA all the above requirements remain valid, but the peculiarity of computations related to VVUQ impose additional needs e.g. for dynamic launching of new jobs in the same or a new allocation, or for handling large amounts of data produced by executed replicas. All of these requirements are addressed in the VECMA architecture within the scope of the Execution layer as it is presented in Figure 5. Starting from the bottom of the diagram, the Execution layer is constituted of three main elements:

- Infrastructure which consists of physical (petascale and smaller) resources and software allowing simulation of next-generation pre-exa- and exascale computational resources;
- Infrastructure Services, which are an abstraction over low-level Infrastructure and provide various means for using easily and more efficiently his Infrastructure;
- A set of automation & execution user-level tools, ranging from Software Development Kit (SDK) libraries, through command-line programs, to graphical and web tools, that enable access to the Infrastructure services for end-users and components of the VECMA Toolkit layer.

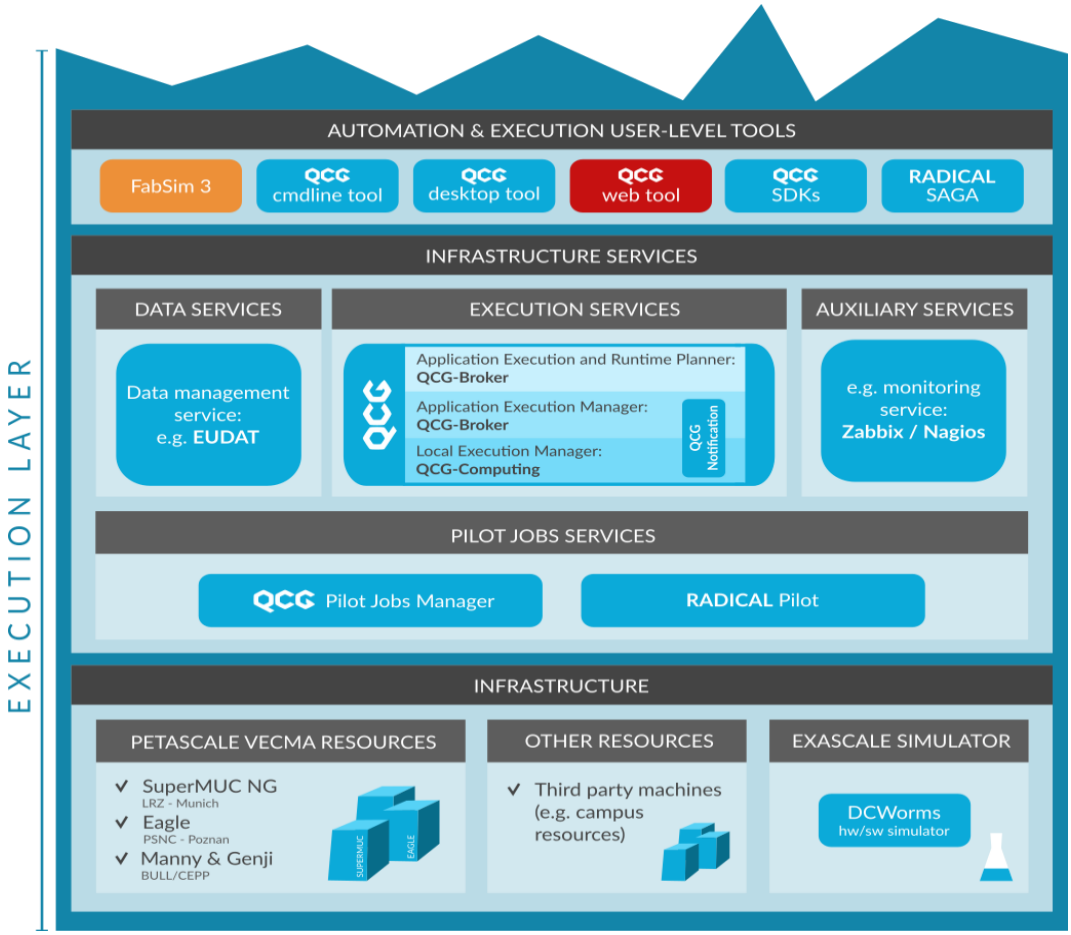


Figure 5 Architecture of the Execution layer

2.4.1 Automation and execution user-level tools

The access to the Infrastructure Services is realised with a group of Automation & execution tools and libraries covering diverse needs and preferences of VECMA users. The usage scenarios are twofold: first, it is expected that the users will directly use the tools, then in some circumstances, the users will employ the tools indirectly using other software components. In the latter case, the specific VECMA VVUQ solutions for users will be built upon the common capabilities of the automation and execution tools and/or libraries. Below we discuss the main features of the particular tools with descriptions of their current and planned roles in the VECMA system.

2.4.1.1 FabSim3

FabSim3 is a software tool for automating computational research activities [10]. It is able to curate input and output data, capture complex workflows in simple one-liner commands, and facilitate job submission to remote hosts directly, as well as to QCG Broker. FabSim3 is written in Python3 and resides in user-space, requiring no administrative privileges to install. It supports a range of plugins, allowing users to modify and extend its functionalities to improve support for a specific application. In

addition, it features a rich configuration environment that is able to capture a wide range of application-specific and environment-specific information in a systematic manner. FabSim3 is part of the VECMA Toolkit, and will be one of the platforms which will facilitate the execution of VVUQ-enabled simulations.

2.4.1.2 QCG clients

The set of QCG client tools allows users to access Infrastructure and perform calculations on a single or many computational resources in an easy and flexible way [11]. The tools work in conjunction with QCG execution services (described in detail later in the document) covering the actual complexity of the multi-resource computing infrastructure with intuitive and flexible interfaces. Depending on the actual needs and preferences, VECMA users will be able to choose from the three different QCG client tools. Let's describe them briefly.

Command line tool: QCG-Client

QCG-Client is a command line tool with the interface very similar to the interfaces of popular queuing systems (e.g. SLURM, PBS). The tool allows submission of different types of jobs, including complex workflows, parameter sweep tasks and array jobs, on single or multiple clusters. It is expected that QCG-Client, through its integration with QCG Pilot Job Manager, will be used in the process of development and testing applications and VVUQ procedures as well as in production VECMA scenarios interchangeable with FabSim3 (the choice of a specific tool will be left to the users' or developers' preferences).

Desktop tool: QCG-Now

QCG-Now[6] is a desktop graphical program that can be easily installed on laptops or PCs and used to run computational tasks on the infrastructure accessible with QCG services. QCG-Now takes care of data transmission, monitoring of execution as well as the security configuration; thus, it may be used even by researchers who don't have proficient IT knowledge. In VECMA QCG-Now is tailored to straightforward execution of typical, predefined computational use-cases.

[6] http://www.qoscosgrid.org/qcg-now

Web tool: QCG-Web

QCG-Web is software which does not yet exist. As its name suggests, QCG-Web will be a web client for QCG middleware. Its actual functionality will be determined in the second part of the project according to the needs of the VECMA exemplar applications, but the initial plans are to build it upon the widely used Jupyter Notebook technology in order to guarantee a good level of flexibility and adaptability for diverse VVUQ workflows.

2.4.1.3 QCG Software Development Kits

The access to QCG execution services can be realized without client tools using APIs. At this moment there are Python and Java SDK libraries available that can be easily exploited to allow access to QCG from previously developed software. This is also the case in VECMA, where QCG SDKs are planned to be used from VVUQ Toolbox, e.g. from EasyVVUQ.

2.4.1.4 RADICAL-SAGA

RADICAL-SAGA is the infrastructure access layer within the RADICAL Cybertools stack[7]. SAGA (Simple API for Grid Applications) defines a high-level interface to the most commonly used distributed computing functionality. It provides an access-layer and mechanisms for distributed infrastructure components like job schedulers, file transfer and resource provisioning services. RADICAL-SAGA, as a lightweight interface, provides standards-based interoperable capabilities to the most commonly used functionalities required to develop distributed applications, tools and services. It is implemented in form of flexible adaptor architecture. Adaptors are dynamically loadable modules that interface the API with different middleware systems and services. In VECMA we will explore the use of RADICAL-Pilot and RADICAL-SAGA in conjunction with FabSim3 and/or specific applications to accelerate the execution of VVUQ scenarios. Infrastructure services

2.4.2 Infrastructure services

The critical requirements of exemplar Applications for efficient VVUQ mechanisms must be reflected in the innovative capabilities of Infrastructure-level middleware services. Cutting-edge multiscale applications, such as those being developed in the VECMA project, where a single run typcially requires thousands of parallel processes, using petascale (or in the near future, exascale) resources. Furthermore, there is a need for numerous such runs in order to sample the relevant phase space in

[7] https://radical-cybertools.github.io/index.html

VVUQ analyses. Thus successful execution of VVUQ analysis within VECMA is predicated on appropriate scheduling and optimized execution of calculations.

When designing the VECMA system it was quickly discovered that it is not generally sufficient to treat the VECMA application executions as standard batch jobs. VECMA application jobs, in contrast to traditional jobs, often dynamically evolve during execution and start new processes as well as alter its requirements. Thus the middleware should not be a hardened black-box, but rather a flexible solution actively cooperating with the architecture layers above. Taking these facts into account we have proposed a hybrid design following the multi-level scheduling approach and consisting of two main modules: Execution Services which are responsible for optimized assignment of job allocations on resources and the Pilot Jobs module responsible for efficient execution of subordinate jobs inside those allocations. In this section we will focus on both these modules, but additionally we will briefly present a set of the supplementary Infrastructure services planned for inclusion in the VECMA Execution layer, e.g. for data management and Infrastructure monitoring.

2.4.2.1 Execution services: QCG

The Execution services are an intermediary component between user-level tools and Infrastructure. In VECMA, they are instantiated by a set of QCG services. Thanks to the selection of QCG to this role it was possible to offer VECMA users a unified and efficient remote access to the underlying infrastructure composed of one or many computing resources of various size, operational policies and local resource management systems. [11]

The QCG middleware consists of three main modules: **Application Execution and Runtime Planner, Application Execution Manager** (both realised as internal parts of QCG-Broker service) and **Local Execution Manager** (realised by QCG-Computing service). The role of Application Execution and Runtime Planner, is to construct the allocation plan assigning parts of the application to the resources, in a way that allows the application to be run in the most efficient way in terms of energy usage and computation time metrics [3]. Application Execution Manager initiates and orchestrates multi-resource computations and manages the whole process of execution of the computational experiment including remote submission and data transfers. Local Execution Managers are deployed on resource provider sites and they are responsible for local execution of jobs in queuing systems installed on resources. Communication between Application Execution Manager and Local Execution Manager is supported by a dedicated QCG publish-subscribe service, namely QCG-Notification.

The formalised VECMA application specification, including VVUQ procedures and requirements, as well as application input files are the basic items that need to be handled by QCG in order to perform VECMA analysis.

2.4.2.2 Data services

The size of data generated (but also consumed) by applications has a critical impact on the achievement of a user's research objectives. This is also true for the VECMA use cases, where each of single-scale application's modules may produce enormous amounts of data (many terabytes, petabytes and more in the future). The way this data should be transferred, how to ensure its confidentiality, integrity and availability, are all issues of concern for the VECMA project. Within the framework of the project, it is necessary to develop procedures and solutions suitable for data transfer as well as long-term data storage, preservation and reuse. In all cases, planning is based on existing and proven solutions developed by European projects and initiatives, in particular EUDAT[8]. A detailed description of data management issues is described in Deliverable D1.3 [12].

2.4.2.3 Auxiliary services

Auxiliary services are not responsible for providing key VECMA functionality, but support the correct and faultless operation of core services. As part of the deployment and maintenance work of the WP5, it is planned, where possible and justified, to use solutions ensuring high availability of created and offered services. The solution of the first choice considered in the project is the Kubernetes[9] platform, which is a system for automating deployment, scaling and management of containerized services. Additionally, in order to ensure the possibility to diagnose the system as a whole, the solutions for aggregation of logs from various components will be deployed and used. In the case of a complex IT system, consisting of many potentially geographically dispersed services cooperating with each other, monitoring becomes crucial for detecting failures and ensuring an appropriate level of stability and availability of the system. For this reason an application of monitoring and alerting services, such as Zabbix[10] or Prometheus[11], is considered for both the traditional and containerized solutions. Additional or alternative tools will be studied throughout the project as potential candidates to increase the overall availability of the VECMA services.

[8] https://eudat.eu/

[9] https://kubernetes.io/

[10] https://www.zabbix.com

[11] https://prometheus.io

2.4.2.4 Pilot Jobs

One of the most important requirements recognized for the VECMA system that is necessary to efficiently process VVUQ analysis is a robust support for execution of large numbers of concurrent executions of a whole application or some of its constituent parts. However, a direct submission of a large number of jobs into a scheduling system can result in long aggregated time to completion as each single job is scheduled independently and waits in a queue. Moreover, the submission of numerous jobs can be restricted or even forbidden by administrative policies defined on clusters. A solution to this problem, that is well suited to the VECMA project, is the employment of the second-level scheduling Pilot Job mechanism. As presented in Figure 6, the Pilot Job can be seen as a regular queueing system job inside which a number of subordinate computing jobs are executed. In Pilot Jobs, a small portion of the allocation is reserved for a service which manages subordinate jobs and offers users a remote interface similar to the interface of a regular queuing system. This hierarchical structure gives a user easy ways to define the most favourable application execution scenarios in terms of efficiency and flexibility.

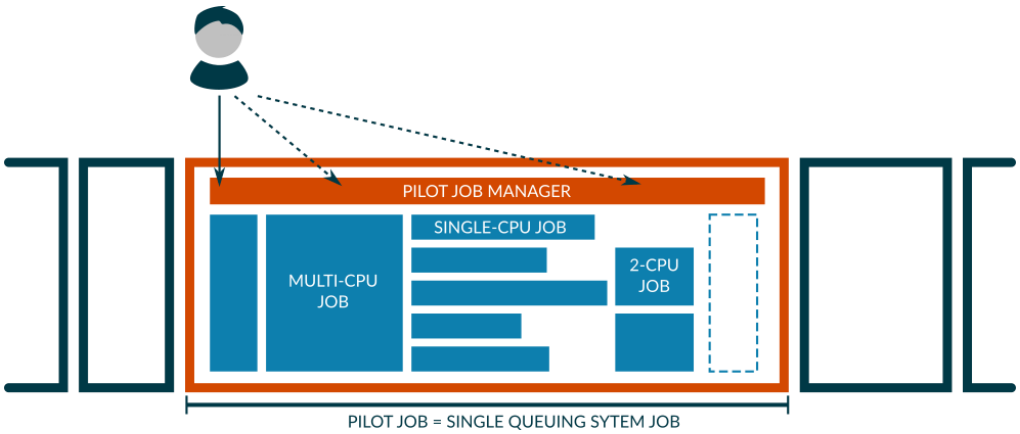


Figure 6 Two-levels scheduling approach in a Pilot Job concept

In VECMA we will use two complementary Pilot Job solutions: QCG Pilot Job Manager and RADICAL-Pilot covering specific requirements of individual VECMA applications. Both of them offer different levels of ease of use and flexibility as well as the possibility of their logical and technical integration with other components of the VECMA system.

QCG Pilot Job Manager

The first implementation of Pilot Jobs to be employed in VECMA is QCG Pilot Job Manager[12] developed by Poznan Supercomputing and Networking Center. Since this solution boils down to a single lightweight service that can be easily instantiated as part of a regular queuing system job and then steered with predefined execution file or REST interface, it may be effortlessly employed by users and integrated with other components. The QCG Pilot Job Manager supports both static scenarios, in which the number of tasks is known in advance, and also dynamic ones, in which additional tasks can be spawned during the life of applications. Dedicated APIs are provided for both use cases. In the VECMA project the current QCG Pilot Job mechanism will be extended to support a global queue and execution of tasks in many allocations. In order to fulfil the needs for efficient and flexible execution of VVUQ tasks and complex application scenarios, it will be integrated with other VECMA components and possibly used by some VECMA applications. The work in progress Is to enable the use of QCG Pilot Job Manager in conjunction with EasyVVUQ. The proposed integration scheme for this scenario is shown in Figure 7.

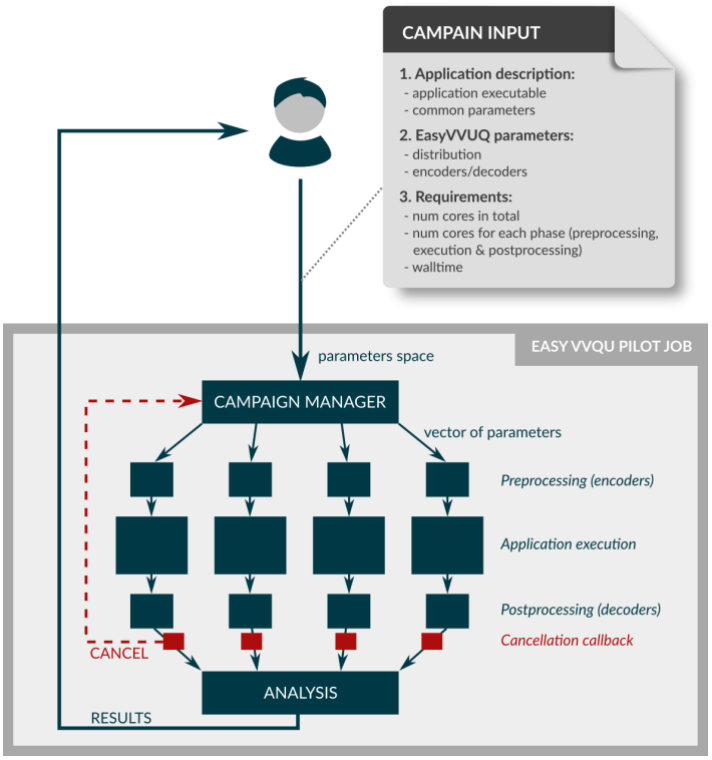


Figure 7 The proposed use of QCG Pilot Job Manager to execute EasyVVUQ tasks

[12] https://github.com/vecma-project/QCG-PilotJob

RADICAL-Pilot

RADICAL-Pilot[13] is a scalable and flexible pilot system. A pilot is a job submitted to a machine in order to acquire exclusive use of a chunk of its resources. Once the pilot becomes active, it can execute the tasks specified by the user. By supporting application-level resource management, it provides a simple and a scalable approach for executing many concurrent simulations and their data requirements on clusters, grids, and clouds. RADICAL-Pilot is written in the Python programming language, and allows user-level control of Pilots and supports a wide range of application types. As a part of VECMA Tool, it will be integrated with other components into FabSim3.

2.4.3 Production and experimental infrastructure

The VECMA testbed is based on two kinds of resources. The first group comprises of large computing clusters of PRACE Tier-0 and Tier-1 class and it is the main platform for both testing and production executions of VECMA scenarios. The second group of resources is composed of several small campus clusters that are used by selected partners to streamline the development process of some VECMA modules. The first phase of the project assumes the use of these traditional HPC resources managed through the queuing system, however in the second phase of the project we will evaluate the possibilities of using cloud environments and a container-based approach. Additionally, in the second phase of the project, in order to evaluate applicability of the developed methods for exascale problems and estimate performance of the constructed software for execution of highly demanding applications on future resources, the VECMA system will use Data Center Workload and Resource Management Simulator - DCWorms [13].

2.4.3.1 Production petascale resources

The hardware resource pool of the VECMA project consists of several powerful machines provided by the project's partners. Namely:

- SuperMUC and SuperMUC-NG at LRZ. SuperMUC is a PRACE Tier-0 system and is in the process of being replaced with SuperMUC-NG, which is currently in the top 10 most powerful computers in the world. SuperMUC NG provides two types of nodes - thin and fat, with the difference being in the amount of memory available. Thin nodes have 96 gigabytes per node, while fat nodes have 768 gigabytes per node. There are 6336 thin nodes with a peak

[13] https://github.com/radical-cybertools/radical.pilot

computational power of 26.3 Pflop/s and then there are 144 fat nodes with a peak computational power of 0.6 Pflop/s. The nodes are all based on the Intel Skylake CPU.

- The Eagle cluster installed at PSNC is a PRACE Tier-1 liquid cooled system managed by SLURM. It consists of 1233 nodes equipped with Intel Xeon E5-2697v3 processors with a total number of 32984 cores. The nodes differ in amount of installed memory, and they are available in 64, 128 and 256 GB configurations. Additionally, Eagle has 3 nodes with two Nvidia V100 GPGPUs each. The network interconnections are realised by InfiniBand FDR. The total computing power of Eagle reaches 1372,13 TFlop/s.
- CEPP at Atos Bull provides access to two supercomputers, Manny and Genji. Manny is a large-scale system, while Genji is a medium-scale system, offering 10 nodes equipped with 32 cores Skylake Xeon processors and 192 GB of memory.

The SuperMuc and Eagle installations are shown in Figure 8.

Figure 8 Supercomputers for use by VECMA: SuperMUC at LRZ (left) and Eagle at PSNC (right)

2.4.3.2 Exascale hardware / software simulator (DCWorms)

Data Center workload and resource management simulator (DCworms) [13] is a simulation tool developed by Poznan Supercomputing and Networking Center (PSNC). DCworms enables modelling and simulation of computing infrastructures to estimate their performance, energy consumption, and energy efficiency metrics for diverse workloads and management policies.

The main goal of DCworms is to enable researchers to evaluate various resource management policies in diverse computing environments. To this end, it supports flexible definition of simulated resources both on physical (computing resources) as well as on logical (scheduling entities) level. This flexible approach allows modelling of various computing entities consisting of compute nodes, processors and cores. Due to an extensible description, users are able to define a number of experiment-specific and visionary characteristics and hardware configurations. DCworms supports simulation of a wide scope of physical and logical architectural patterns that may span from a single computing resource up to whole data centres (even geographically distributed).

The VECMA project will examine the usefulness and practical applicability of the DCworms simulator to optimise the use of resources mostly in terms of time and energy consumption in the process of determining the uncertainty of results of multi-scale applications for existing computing infrastructures available in the project and potential future exascale resources.

3 Conclusions

The architecture presented in this deliverable addresses all the needs identified so far in the project and has been approved by the consortium for realization of the VECMA system. The separation of concerns defined here through the Application, VECMA Toolkit and Execution layers reflects the distribution of responsibilities among three technical Work Packages. The architecture described herein is a reference position for VECMA and will be implemented, deployed and verified in the coming phases of the project. Nevertheless, when requirements or conditions change, the project's architecture design will be adapted where necessary.

It is expected that the compiled list of fast track components, which consists mainly of already existing software, and the proposed means of functional and technical integration, will allow us to build basic VECMA environment no later than at the end of the first year of the project. In later stages of the project—by extending this environment with deep track components—the VECMA functionality will be iteratively enhanced so as to ultimately attain a mature environment for VECMA users to exploit.

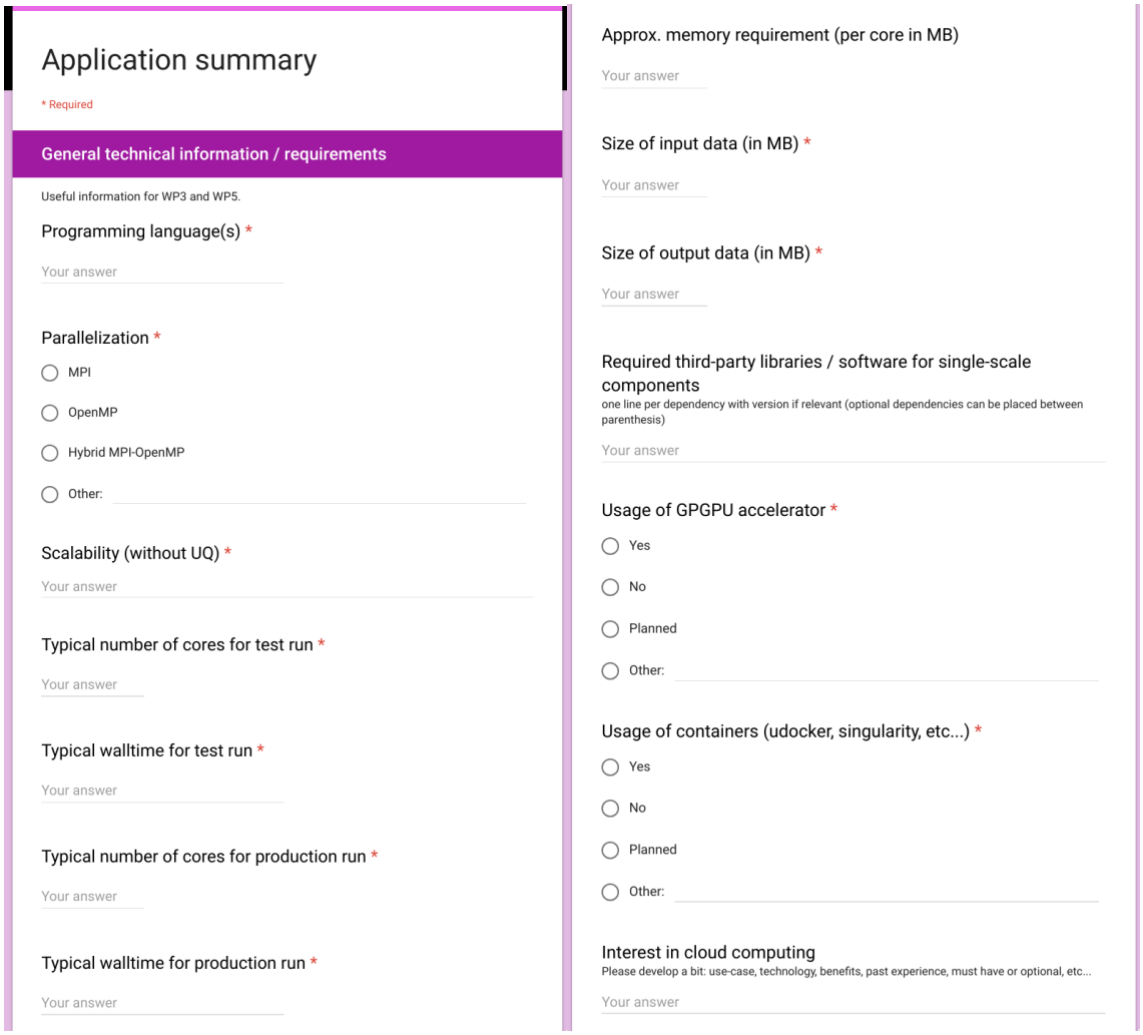
4 Annexes

Annex 1. Software components used in VECMA

	Component	Fast track	Deep track	Origin	Development plans
Applications layer	**FabSim3**	see FabSim3 in the Execution layer			
	QCG-Client	see QCG-Client in the Execution layer			
	MUSCLE2	✓		MAPPER, ComPat	Use for developing surrogate models, no further development.
	MUSCLE3		✓	e-MUSC	Implementation from scratch, integration with FabSim3 and/or QCG Pilot Job Manager.
	OMUSE	✓		ABC-MUSE	Use for developing surrogate models, no further development.
VECMA Toolkit layer	**Verification & Validation Patterns (VVP)**	✓		VECMA	Implementation from scratch
	Uncertainty Quantification Patterns (UQP)	✓		VECMA	Implementation from scratch
	VVUQ Composition: EasyVVUQ	✓		VECMA	Implementation from scratch
	Execution Description Builder: FabSim3/QCG tools	see FabSim3/QCG tools in the Execution layer			
Execution layer	**FabSim3**	✓		CRESTA, ComPat (FabSim1)	Extensive redevelopment, with focus on more flexibility, improved scalability, and more support for external tools and applications.
	QCG Cmdline tools: QCG-Client	✓		QosCosGrid, MAPPER, ComPat	Integration with QCG Pilot Job Manager

	QCG Desktop tools: QCG-Now		✓	PLGrid	Support for execution of VVUQ tasks (e.g. EasyVVUQ in QCG Pilot Job Manager)
	QCG Web tools: QCG-Web		✓	VECMA	Implementation from scratch
	QCG SDKs	✓		PLGrid	Minor updates
	RADICAL SAGA			US NSF ACI 1440677 and ACI 1253644	Support, minor updates and integrations.
	QCG: Application Execution and Runtime Planner	✓		QosCosGrid, MAPPER, ComPat	Minor updates
	QCG: Application Execution Manager	✓		QosCosGrid, MAPPER, ComPat	Integration with QCG Pilot Job Manager
	QCG: Local Execution Manager	✓		QosCosGrid, MAPPER, ComPat	Support for execution EasyVVUQ in QCG Pilot Job Manager
	QCG Pilot Jobs Manager	✓		ComPat	Support for global queue, execution of tasks in many allocations. integration with other VECMA components, e.g. EasyVVUQ.
	RADICAL Pilot			US NSF ACI 1440677 and ACI 1253644	Support, minor updates and integrations.
	Exascale hw/sw simulator (DCWorms)		✓	CoolEmAll	Adaptation to the multiscale scenarios

Annex 2. WP5 related part of the Applications Questionnaire



Application summary

* Required

General technical information / requirements

Useful information for WP3 and WP5.

Programming language(s) *

Your answer

Parallelization *

- MPI
- OpenMP
- Hybrid MPI-OpenMP
- Other:

Scalability (without UQ) *

Your answer

Typical number of cores for test run *

Your answer

Typical walltime for test run *

Your answer

Typical number of cores for production run *

Your answer

Typical walltime for production run *

Your answer

Approx. memory requirement (per core in MB)

Your answer

Size of input data (in MB) *

Your answer

Size of output data (in MB) *

Your answer

Required third-party libraries / software for single-scale components

one line per dependency with version if relevant (optional dependencies can be placed between parenthesis)

Your answer

Usage of GPGPU accelerator *

- Yes
- No
- Planned
- Other:

Usage of containers (udocker, singularity, etc...) *

- Yes
- No
- Planned
- Other:

Interest in cloud computing

Please develop a bit: use-case, technology, benefits, past experience, must have or optional, etc...

Your answer

Number of single-scale components in the multiscale application *

Choose

Components instantiation *

- ○ Static
- ○ Dynamic
- ○ Other:

Coupling software *

- ○ Fabsim3
- ○ MUSCLE2
- ○ OMUSE
- ○ QCG Pilot-Job
- ○ Scripts
- ○ Other:

Type of coupling *

- ○ one-way workflow
- ○ two-ways workflow (feedback)
- ○ loop
- ○ Other:

Type of data being transferred between coupled components *

- ☐ files
- ☐ simple data types
- ☐ structures / objects
- ☐ byte stream
- ☐ Other:

Size of data being transferred between coupled components (in MB) *

Your answer

Specific technical requirements (open ports, etc...)

Your answer

New development/software needs/wishes

You can describe here all the remaining needs and wishes (in terms of non-UQ software) for your application: it can concern instantiation, deployment, scheduling, etc... This could help WP3 to steer the developments, but does not automatically become a feature request for the toolkit.

Your answer

5 References

[1] Deliverable D4.1 - Report on application software readiness (M9)

[2] Borgdorff, J., Falcone, J. L., Lorenz, E., Bona-Casas, C., Chopard, B., & Hoekstra, A. G. (2013). Foundations of distributed multiscale computing: Formalization, specification, and analysis. *Journal of Parallel and Distributed Computing*, **73(4)**, 465-483

[3] Alowayyed, S., Piontek, T., Suter, J. L., Hoenen, O., Groen, D., Luk, O., ... & Brabazon, K. (2019). Patterns for high performance multiscale computing. *Future Generation Computer Systems*, **91**, 335-346.

[4] Borgdorff, J., Mamonski, M., Bosak, B., Kurowski, K., Belgacem, M. B., Chopard, B., ... & Hoekstra, A. G. (2014). Distributed multiscale computing with MUSCLE 2, the multiscale coupling library and environment. *Journal of Computational Science*, **5(5)**, 719-731.

[5] Pelupessy, I., Van Werkhoven, B., Van Elteren, A., Viebahn, J., Candy, A., Zwart, S. P., & Dijkstra, H. (2017). The oceanographic multipurpose software environment (omuse v1. 0). *Geoscientific Model Development*, **10(8)**, 3167.

[6] Pelupessy, F. I., van Elteren, A., de Vries, N., McMillan, S. L. W., Drost, N., & Zwart, S. P. (2013). The astrophysical multipurpose software environment. *Astronomy & Astrophysics*, **557**, A84.

[7] Deliverable D2.1 - Report on multiscale UQ algorithms based on non-intrusive MC and semi-intrusive MC and mapping thereof in UQPs (M12)

[8] Deliverable D2.2 - Report on advanced multiscale UQ algorithms, including intrusive approaches, and mapping thereof in UQPs and first results on V&V (M24)

[9] Deliverable D3.1 - Report on the fast-track implementation of the VECMA toolkit (M9)

[10] Groen, D., Bhati, A. P., Suter, J., Hetherington, J., Zasada, S. J., & Coveney, P. V. (2016). FabSim: Facilitating computational research through automation on large-scale and distributed e-infrastructures. *Computer Physics Communications*, **207**, 375-385.

[11] Piontek, T., Bosak, B., Ciżnicki, M., Grabowski, P., Kopta, P., Kulczewski, M., ... & Kurowski, K. (2016). Development of Science Gateways Using QCG—Lessons Learned from the Deployment on Large Scale Distributed and HPC Infrastructures. *Journal of Grid Computing*, **14(4)**, 559-573.

[12] Deliverable D1.3: Data Management Plan (M6)

[13] Kurowski, K., Oleksiak, A., Piątek, W., Piontek, T., Przybyszewski, A., & Węglarz, J. (2013). DCworms–A tool for simulation of energy efficiency in distributed computing infrastructures. *Simulation Modelling Practice and Theory*, **39**, 135-151.