D3.4 Full release of the entire VECMA toolkit

Due Date	14 June 2021
Delivery	07 June 2021
Submission of updated version	N/A
Lead Partner	UBRU
Dissemination Level	Public
Status	Final
Approved	Executive Board
Version	V1.2

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
27.05.2021 v1.0	Diana Suleimenova	First Draft
	Derek Groen	
	Hamid Arabnejad	
28.05.2021 v1.0	Xuanye Gu	Sent for internal review
4.06.2021 v1.1	Derek Groen	Revised after receiving internal reviewers' feedbacks
	Diana Suleimenova	
5.06.2021 v1.1	Peter Coveney	Sent final comments
7.06.2021 v1.2	Diana Suleimenova	Finalised report

CONTRIBUTORS

- Diana Suleimenova – Main Author, Editor
- Derek Groen – Author
- Hamid Arabnejad – Author
- Bartosz Bosak – Reviewer
- Wouter Edeling – Reviewer
- Vytautas Jancauskas – Reviewer
- Peter Coveney – Approver, Editor

Disclaimer

Table of Contents

1 Executive Summary

This deliverable is for Work Package 3, listed in the 800925_Annex 1- Description of the action (Part A) file. As this is a software deliverable, its contents are available online rather than in this document. However, in this written document we summarize the user review feedback we received and how we incorporated it in the final release.

We conducted a total of 12 user surveys, and obtained a total of 107 in-depth responses over the course of the project. The feedback we obtained was constructive in all cases, and frequently led us to resolve critical issues in the toolkit. Additionally, we believe the frequent and in-depth interaction with users have contributed to the relatively high user uptake of the tools compared to our expectations, which was already manifest during the stage where the software was only available in preliminary form.

A recurring theme in the user feedback was the ease of deployment and the accessibility of good documentation, and we have made considerable updates in both areas throughout this project. In addition, we found that users across diverse domains use our toolkit, believing it has clear added value in terms of VVUQ, and that many users indeed use the VECMA toolkit (VECMAtk) at scale, with the support of supercomputing resources.

The VECMA toolkit Month 36 release is available at: https://www.vecma-toolkit.eu/.

2 Introduction

This deliverable consists of four parts. In Section 3 we briefly describe the M36 VECMAtk release. In Section 4 we present how we gathered user feedback over the lifetime of the project, and in Section 5 we describe how we incorporated that feedback in our development. We then share our main findings and insights in the concluding Section 6.

3 The final VECMA toolkit release

Prior to this deliverable, the VECMA toolkit has had two major releases (M12 and M24), which are reported in D3.2 and D3.3. In addition, we have released incremental updates in M6, M9, M15, M18, M21, M27, M30 and M33.

As of M36, the toolkit consists of the following components:

- EasyVVUQ
- EasySurrogate (introduced in M30)
- FabSim3
- QCG-PilotJob
- MUSCLE3 (developed in collaboration with the Dutch e-Science Center)
- QCG-Client
- QCG-Now

Compared to the M33 and M24 releases, one component has been removed, namely the EasyVVUQ-QCGPJ Integration API, or EQI. This component is no longer necessary as the M36 release now features a direct integration between QCG-PilotJob and EasyVVUQ.

Full documentation of the toolkit and the components is beyond the scope of this written deliverable, which is oriented towards reporting user feedback. However, all the necessary information can be found at https://www.vecma-toolkit.eu/.

Although M36 is labelled as the “final” toolkit release, we will be incorporating further improvements to the software and documentation up to the now extended end of the project (M42). As part of those activities, we will schedule an additional updated release in M42.

4 User feedback elicitation

Prior to the start of VECMA, it was not completely clear in what way we would collect feedback from the users of the toolkit. However, we did have working software from the first month of the project, and given our preference for evolutionary prototyping, we realized that it was crucial to obtain user feedback early on.

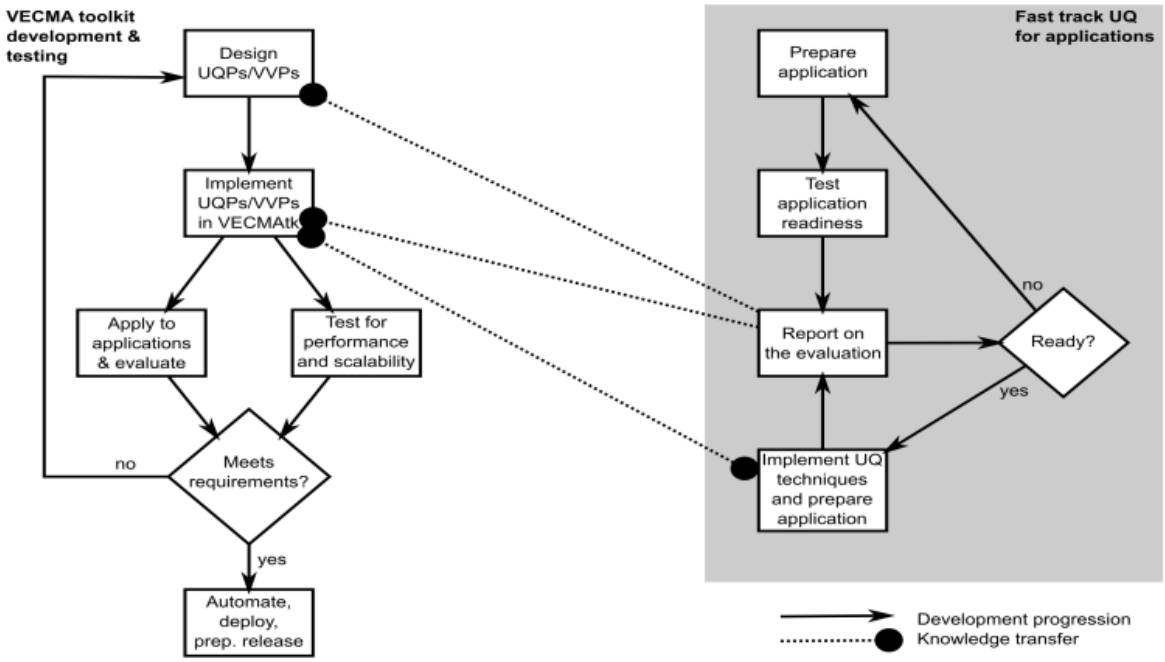


Figure 1: Diagram showcasing the evolutionary prototyping approach in the original VECMA proposal. Here, application-specific fast-track work was intended to be done in Year 1, in the process informing deep track toolkit development that would mainly come to fruition in Years 2 and 3.

Early on in the project, the toolkit adoption was limited (most users did manual UQ on their applications), and we chose to solicit user input using a Google Forms survey. Our first user survey in M3 solicited only 3 responses due to the small userbase, but was otherwise successful in soliciting user feedback (See Appendix A). Based on this, we then decided to do an internal release every 3 months, and have that accompanied with a so-called Alpha User survey.

Here, alpha users were considered persons that were willing to test out the VECMA toolkit and, where relevant, try to apply specific routines to their own research applications. The response rates were above 50% during the first year, but started to tail off after M12 as the user base grew further (~+1 new alpha user per month). The main (verbally provided) reason given by the users was the repetitive

nature and length of the questionnaires, and the fact that many of them actually didn't reinstall components every 3 months, so they felt their feedback was not that relevant.

In M18 we experimented with a different approach, splitting the alpha users into Testing users and Research users, making a separate questionnaire for each of them. Though initially successful with 11 responses, the response rate quickly dropped again after M21, forcing us to rethink our approach once more.

In M27 we therefore developed the last, shorter, consolidated feedback form. This only led to a mild increase in response rate as we realized that the user base clearly was subject to survey exhaustion by this time.

In the table below we summarize the number of responses from each of these regular surveys, and a brief highlight.

Month	# of responses	comments
3	3	Covered only FabSim3, main response was about lack of clarity about the tool's purpose and lack of documentation.
6	8 (from recollection)	Results lost due to accidental overwrite when developing the M9 survey.
9	6	Included also QCG-Client and EasyVVUQ. Main response was to have more documentation for QCG and to have more examples for EasyVVUQ. Also, strong suggestions (expressed during a meeting, not in the survey) to create Docker / Singularity versions for portability.
12	10	Included also QCG-PilotJob and QCG-Now. Main response was low actual uptake of containerized versions (1 out of 10 installed in this way). Clear criticism on the FabSim3 plugin installation procedure, which involved using `git clone`. M12 feedback was given *prior* to the official M12 release.
15	6	Included also MUSCLE3. Main response was again requests for better documentation, and lack of uptake of MUSCLE3 and QCG-Now (at the time).
18	7 (research) + 4 (testing)	Separate research and testing surveys. Main response was a request to add adaptive sampling and to facilitate more pip-based installation routines.
21	10 (research) + 5 (testing)	Clear evidence of uptake at scale. Single survey with most valuable responses in the project. Increase in uptake of MUSCLE3. Reasonably short installation duration times.

24	2 (research) + 1 (testing)	Clear evidence of survey exhaustion. Results contained little directly useful information.
27	6	Shortened form, which also included EasySurrogate. Main response was on installation issues with QCG-PilotJob, improved documentation for EasyVVUQ, and evidence of research uptake for FabSim3.

After M27 we realized that the periodic survey was losing momentum, and instead opted to do a user survey after each VECMA Hackathon that we organised. Please see the table below for a summary of those:

Month	# of participants	# of responses	comments
31	37	19	Clear evidence of HPC uptake.
34	18	12	Main response was that EasyVVUQ had many inconsistencies, causing documentation to be outdated and examples to be no longer working.
35	15	8	Main response was once more a push for better documentation.

Throughout these three events, approximately 50-70% of the participants filled in the survey, and indeed the drop in responses largely correlates with the drop in participation, as users got a little saturated with the frequency of the Hackathons. We expected this in advance, but nevertheless pushed for a M35 Hackathon because it would give us a clear sense of what to prioritize for this final release.

Though the main result highlights are provided here, either in the table or in the Appendix, there is one graph worth adding here. We asked users how many cores they used with their VECMAtk application in Month 31, which led to the following distribution:

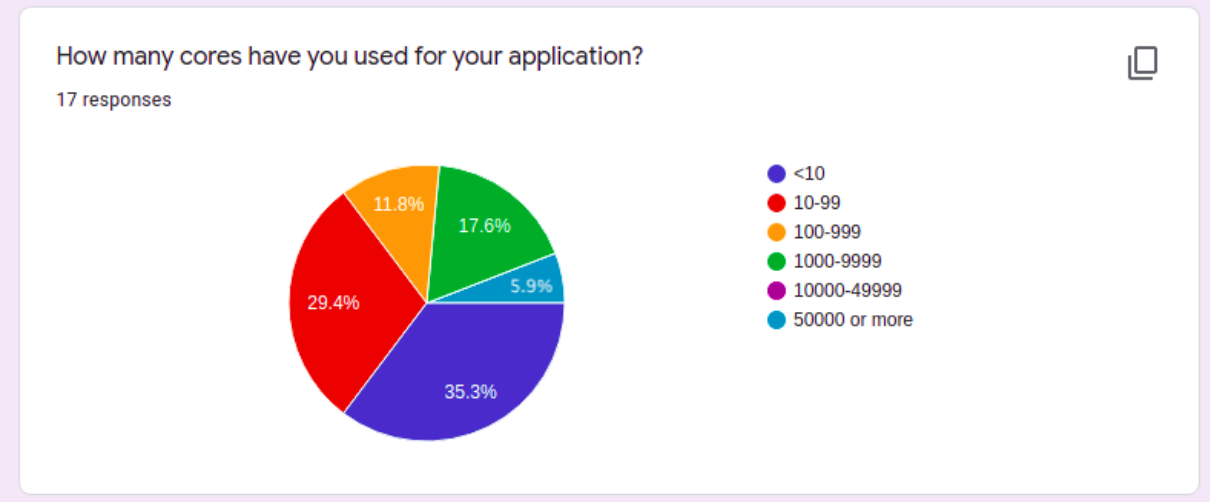


Figure 2. The number of cores used by users for their application with VECMAtk components.

In general, we find that most VECMAtk users operate in the so-called long tail of HPC, but that there are a significant number of users that run applications across multiple nodes. This is also evidenced by the variety of articles that have been published reporting work at that scale, and that have relied on VECMAtk.

We have used the feedback that we received over all these surveys to guide the development process. In many cases this led to immediate improvements, although we did find that issues around deployment and documentation tend to recur whenever a major update to particular components had been made. In the next section we will reflect on this in more detail.

5 *Changes made based on user feedback*

Here we describe the main changes that we have made to the VECMA toolkit, based on user feedback, and reflect on the effectiveness of these changes.

VECMAtk Installation process

The initial public release of VECMAtk in M9 (M6 and M3 were not public yet) included three components, which were FabSim3, EasyVVUQ and QCG-Client. Over time, we added another five components to the toolkit, namely EasySurrogate, MUSCLE3, QCG-PilotJob, QCG-Now and EQI, the latter of which has since been removed again [1]. Originally, the installation of these components was manual and independent from each other which took time and had various dependency issues. In response to this, we collected users' installation preferences, which included *manual, docker or singularity, master build script, pip3 install vecmatk* or *cloning the VECMA repository with everything*. Installation preferences differed at each survey iteration, where some users preferred manual while

others favoured pip3, docker or singularity. However, over time we found there was an overall preference for pip3 installation, once all alternatives were available.

We also received requests from users for an integrated or one-line installation of components. Thus, we implemented a wholesale VECMAtk installation script that contains all components and downloads them at once in the form of a Python VirtualEnv. The documentation of the installation process can be found on the VECMAtk Github repository[1]. This method is provided in addition to the individual installations of the VECMAtk components, which are available as well. To assess the installation time, we asked users to indicate how much time they spent installing software. Over time, we significantly improved and currently it is between 10-30 minutes (on average) for each component (last tested in M27, see Appendix).

VECMAtk Components

In this section, we provide highlights of our progressive improvements and modifications in the VECMAtk components based on users' feedback and remarks.

FabSim3 is a Python-based automation toolkit for scientific simulation and data processing workflows, which has been substantially transformed within the VECMA project. Based on the M3 feedback, we improved the installation process and documentation. This led to favourable comments in the M9 survey, such as "Good documentation makes FabSim3 quite easy to use. Now I'm waiting to use it with real applications to have more experience with it.". During the M12 survey and the hackathons in that period we received criticism on how plugins worked and were installed. Because of that, we implemented a new plugin architecture. We had already introduced a FabDummy plugin that is an example application providing introduction to the FabSim3 functionalities and capabilities, but based on this feedback we also created one-line commands to install plugins using FabSim3, instead of doing "git clone" manually. Later on, we also introduced the ability to make plugins dependent on each other, and to distribute custom (plugin) functionalities across multiple imported files. In addition to the changes described in D3.2[2] and D3.3[3], we also strengthened documentation in response to the recent surveys, redoing the ***readthedocs*** site and adding more tutorials for users as well as for plugin developers. We also received verbal feedback that the inferior pilot job scalability when using FabSim3

[1] https://github.com/vecma-project/VECMAtk
[2] https://www.vecma.eu/wp-content/uploads/2019/10/VECMA_D3.2_Fast-Track-Release-Toolkit_UBRU_V1.0-20190614.pdf
[3] https://www.vecma.eu/wp-content/uploads/2020/09/VECMA_D3.3_Preliminary-deep-track-release-of-the-VECMA-toolkit_UBRU_v1.1_20200612.pdf

was deterring some users, so we introduced a new job submission workflow which reduced the elapsed submission time per simulation instance from seconds to milliseconds.

EasyVVUQ is a Python library designed to facilitate verification, validation and uncertainty quantification (VVUQ) for a wide variety of simulations. According to our questionnaires, EasyVVUQ is the most predominantly used component, and has found higher than expected uptake in various domains ranging from biomedicine, advanced materials, urban air pollution to e.g. fusion energy. To provide support to each of these domains, EasyVVUQ developers improved the component by implementing new functionalities in the framework, providing several sampler methods and integrating EasyVVUQ with other VECMAtk components, such as QCG-PilotJob and EasySurrogate. These improvements are driven by aspirations of developers and application users, as well as suggestions and requests of external users. We state this because responses from users mostly requested new implementations, such as parameter fitting and optimization, and did not have any bug issues or problems when using EasyVVUQ. However, the great demand for additional features took its toll in Year 3, and led to tutorials and documentation to become outdated as a result. Consequently, we have chosen to primarily focus on hardening the software and updating / fixing / improving the documentation for this release.

QCG tools consist of QCG-Client, QCG-PilotJob and QCG-Now, which aim to provide efficient and reliable execution of a large number of computational jobs on the HPC machines. The main distinction between them is that QCG-Client is a command-line interface, QCG-PilotJob is a lightweight implementation of the Pilot Job mechanism and QCG-Now is a portable desktop program. Early on, several users complained about unclear documentation, which was subsequently improved. In addition, QCG-PilotJob found very high uptake among VECMA users, whereas QCG-Client and QCG-Now did not have such widespread use. As a result, the development resources in the project have been gradually diverted towards supporting and improving QCG-PilotJob, with lesser emphasis on the two other tools. Moreover, the VECMA developers simplified the usage of EasyVVUQ with a QCG-PilotJob engine that aimed for efficient and parallel execution of demanding EasyVVUQ scenarios on HPC resources. The integration of EasyVVUQ and QCG-PilotJob, namely EQI, has reached an intermediate stage in its development. As noted above, it has been excluded from the M36 release as EasyVVUQ has now been directly integrated with QCG-PilotJob.

There were initially no major concerns about the QCG tools from our users. However, some users suggested that they struggled to gain access to these tools as a new user and found instructions ambiguous while others battled with the execution of submitted jobs as they were pending and taking

time, which in their experience was inconvenient. Many of these issues were outside of the control of the developers, but they did make significant advances with QCG-PilotJob in terms of performance, stability and documentation[4], which was done to directly address verbally raised concerns amongst HPC users of non-PSNC supercomputers such as SuperMUC-NG, ARCHER2 and Cartesius.

MUSCLE3 is the third incarnation of the Multiscale Coupling Library and Environment, which aims to easily create coupled multiscale simulations and then enable efficient UQ analysis using advanced semi-intrusive algorithms. It was initially released with solid documentation, but with little testing on HPC resources, leading to initial feedback to improve the deployment there. The MUSCLE developers have since improved its compatibility on several operating systems, and made the documentation more thorough. Moreover, users suggested that the divisions in the tutorials provide a clear picture of the component. There were also requests to implement UQ features for multiscale workflows that developers took into account for M36. In addition, the recent Hackathon responses inform that users would like to see the integration of MUSCLE3 with EasyVVUQ and MUSCLE3 with QCG-PilotJob for data interchange and coupling campaigns (the latter is underway). Because MUSCLE3 is externally developed within the Netherlands e-Science Center, not all of these concerns can be immediately addressed. However, they have been noted as development priorities for the tool, and we hope these improvements will make their way into the M42 release.

The **documentation, tutorials and examples** of the VECMAtk components faced iterative and continuous improvements (and sometimes regressions) as we added and modified components with new functionalities and features, integrations and applications. Hence, we received responses that indicated outdated documentation or tutorials for some of the toolkit components and required an overall sequential order to follow the application of components easily. We addressed these remarks after every feedback round and still continue to improve and provide up to date instructions to application and external users. To produce descriptive, comprehensive and consistent documentation, tutorials and examples, we divided them into static, interactive and video tutorials, which are described in D3.5[5]. In turn, it allowed us to target different types of users from various domains. We will also provide additional in-source documentation in the M36 release to interpret functions and avoid plain code fragments. In addition, users suggested to include more tutorials with real applications, which exist for various domain applications [2]. However, users might not have been aware about these application tutorials as they are scattered across repositories. We aim to direct

[4] https://qcg-pilotjob.readthedocs.io

[5] https://www.vecma.eu/wp-content/uploads/2020/09/VECMA_D3.5_Documentation-and-Tutorials_UBRU_v1.2_20200814.pdf

users to the comprehensive guide of all application examples, which can be found at https://www.vecma-toolkit.eu/tutorials/, in the final release.

6 Conclusions

In this deliverable we summarized the VECMAtk release briefly, and more extensively described how we solicited user feedback, and used that feedback to improve the toolkit.

Getting consistent response rates is not easy, and we had to reinvent the feedback process multiple times in order to maintain a reasonable level of user feedback. But we found that this adaptation effort was worth it, as the feedback we obtained was crucial and helpful in numerous cases.

There were a few examples where user feedback was not directly productive: most notably the strong user preference for containerization in Year 1 was followed up with a near non-existent uptake of the containerized version in Year 2. Likewise, we found several examples on tool documentation where user feedback was initially addressed, only for the solution to degrade again over time to the point that users again provided the same criticisms.

Developing a production-level toolkit in the context of a research project was always going to be a challenging endeavour as the resources in such projects are allocated towards research and as such user experience suffers. It is also because we have neither the necessary skills, nor the personnel nor the incentives, but the high uptake of the toolkit certainly was a mixed blessing in that regard. While we very much welcome the major uptake of the toolkit, and the solid number of research publications that have cited it, it also forced us to redirect vast amounts of effort from core development to user support. As an example, EasyVVUQ, originally a UCL-led tool, found such booming uptake that it subsequently received major development effort from CWI (Wouter Edeling), LRZ (Vytautas Jancauskas), IPP (Jalal Lakhlili) and in the most recent times BUL (Hamid Arabnejad). Likewise, development and support efforts at PSNC were heavily concentrated on QCG-PilotJob to reflect the high uptake there. In terms of technical progress, all this switching of roles was far from ideal and led to delays in places, but the upside is that we now have a group of individuals that each have development-level experience with multiple components in VECMAtk. That in-depth expertise across the different tools, and across different people, is likely to help us to sustain the toolkit expertise after VECMA has officially concluded.

7 References

[1] Groen D., et al. 2021 VECMAtk: A scalable verification, validation and uncertainty quantification toolkit for scientific simulations. Phil. Trans. R. Soc. A.3792020022120200221.
DOI: http://doi.org/10.1098/rsta.2020.0221

[2] Suleimenova D., et al. (in press) Tutorial applications for verification, validation and uncertainty quantification using VECMA toolkit. Journal of Computational Science.

8 Appendix A: Highlights from survey results

Month 3 (covering only FabSim, FabMD and FabDummy):

What kind of improvements would you like to see in VECMAtk (and any of its components)?

2 responses

Can't think of anything at this point.

The messages that appear when FabSim are run are very confusing - is there a better way of informing the user of the status of his simulation: where it is running, was it submitted OK, where to expect the results etc. What is the point of the stdout stderr print out etc? Is there an chance of seeing the errors on the remote machine if it fails etc? This are just my first impressions, I will report back with more when I get my application running with VECMAtk.

What kind of added documentation, guides or references would you like to see in VECMAtk (and any of its components)?

2 responses

A separate set of instructions for testing on a remote host e.g. ARCHER would have been nice. I didn't want to install LAMMPS on my laptop.

Perhaps a general breakdown of the file structure, to better understand how the toolkit works and where inputs are taken from, how they are processed, and where the output goes.

Firstly, there is no description of what you can actually do with FabSim, there is a lot of running examples (dummy, for example) with out describing what they can do and what is possible with your own application. I'm not sure where to start in https://github.com/djgroen/FabSim3/tree/master/doc - there are lots of pages there. Some sort of guide to which are important early on etc. Lets take a simple example - how do I run something remotely and then analysis locally?

Month 9 (also covering EasyVVUQ and QCG-Client)

Please describe you experience of using FabSim3

3 responses

Good documentation makes FabSim3 quite easy to use. Now I'm waiting to use it with real applications to have more experience with it.

I really enjoyed it. It is very intuitive and easy to adapt to particular problems. It was definitely key in my research.

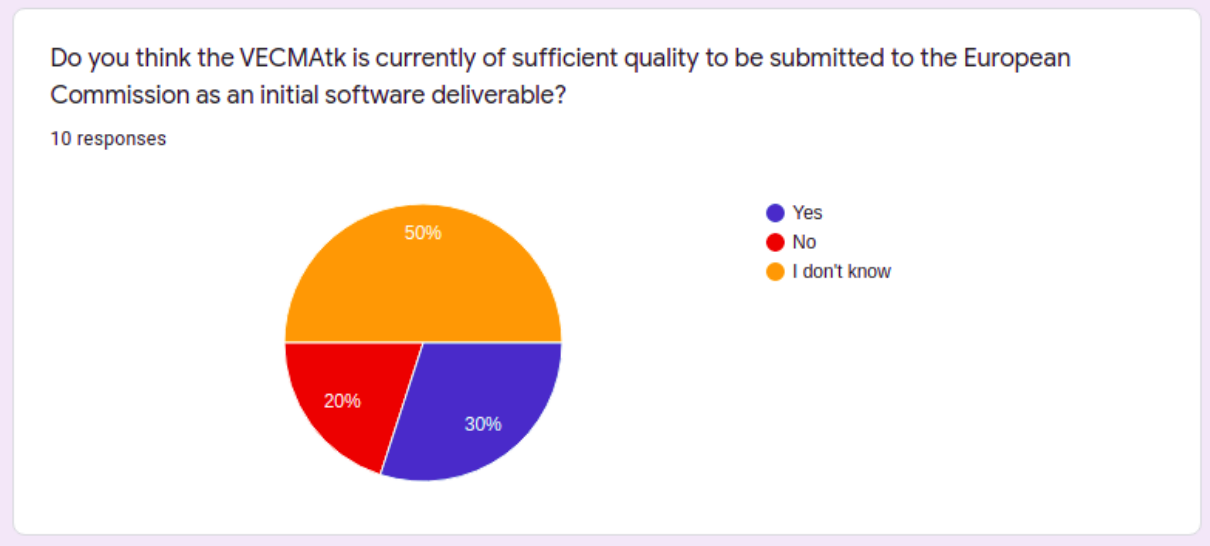
As I direct user of FabSim3, I find FabSim3 pleasant, straight-forward and useful at this stage.

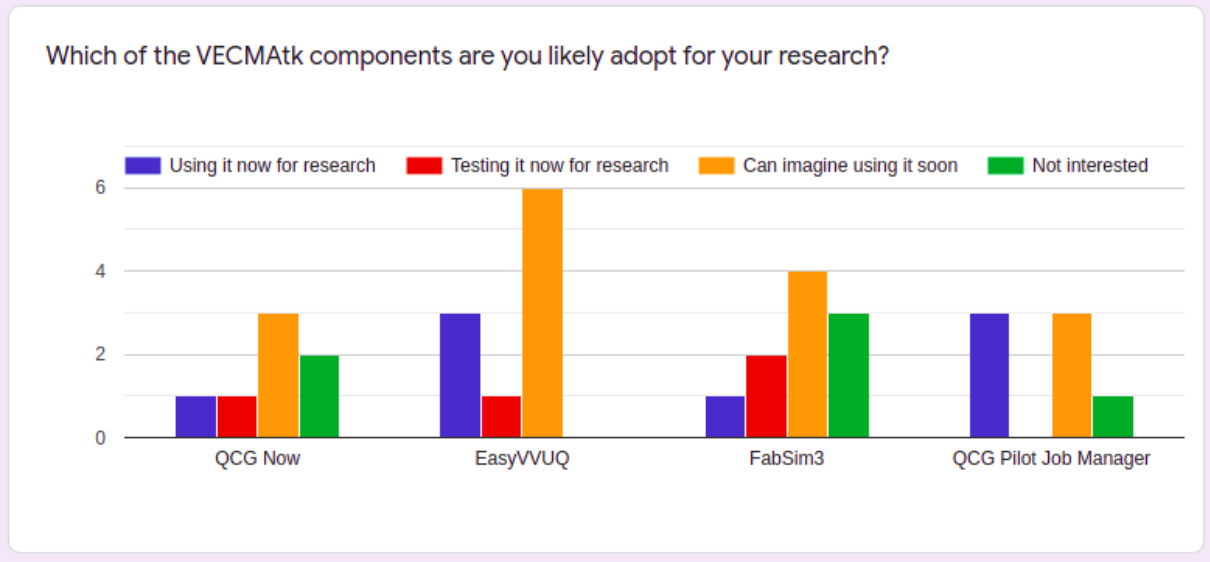
EasyVVUQ

Were you able to integrate EasyVVUQ into your workflow?

6 responses

Month 12 (also covering QCG-PJ and QCG-Now)





Month 15 (also including MUSCLE3)

What kind of added documentation, guides or references would you like to see in VECMAtk (and any of its components)?

3 responses

Detailed documentation on what is available to apply on each component and simpler examples for first time users in a single platform (as first time user has to open all different repositories and tutorials that can be confusing).

As mentioned before, schematic diagram showing how the tools can integrate into workflows will prove to be very useful.

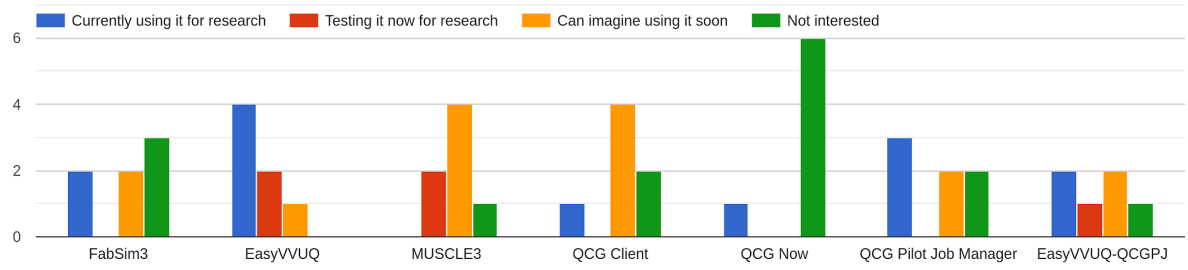
I found that I was only truly able to understand how FabSim3 worked when sitting with Hamid and the help of Derek for an afternoon.

A few simple (possibly interactive) schematics could help drastically.

Clearer examples of using VVUQ for validating simulation data against experimental data, and performing uncertainty quantification on data extracted from an ensemble. Any MD code will do.

Month 18

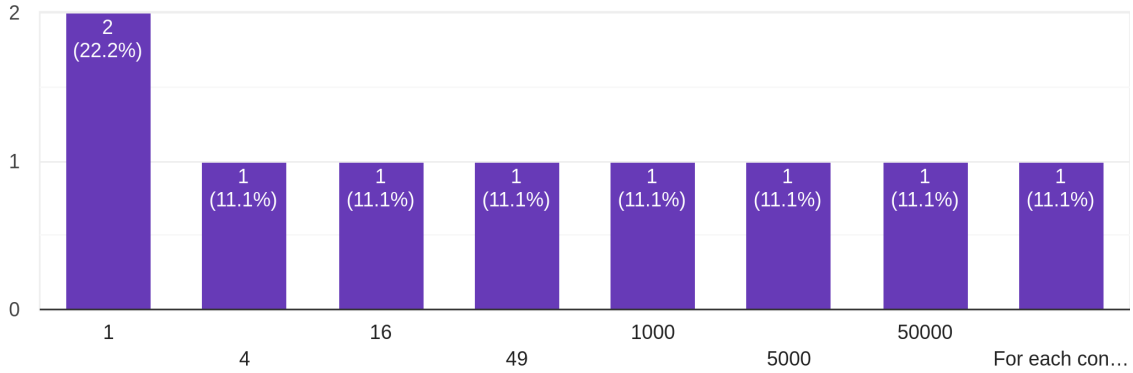
Which components do you use?



Month 21

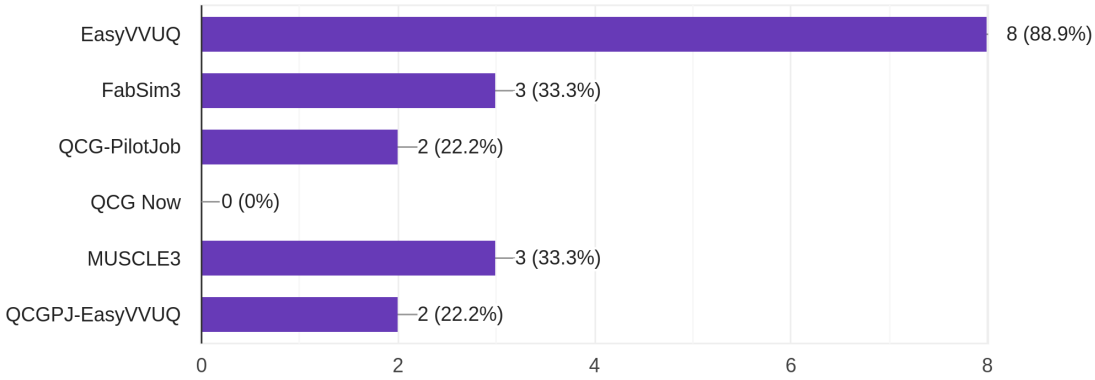
What is the largest number of cores you have used with VECMAtk in one simulation workflow?

9 responses

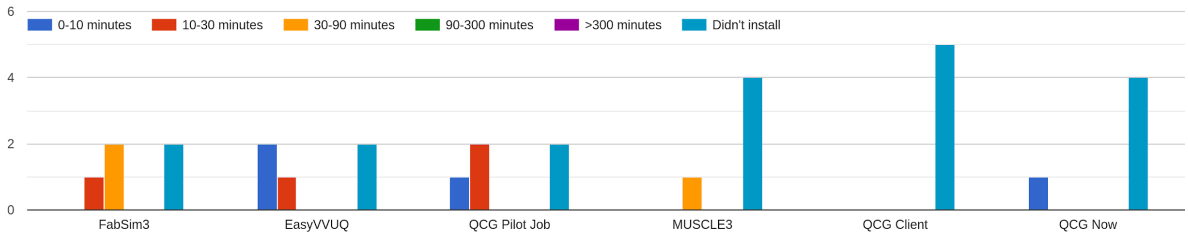


Which VECMAtk components were involved in this particular workflow?

9 responses

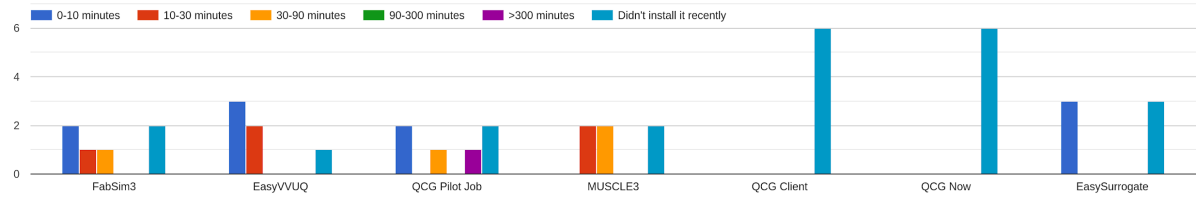


How long did it take you to get each component installed (if applicable) and basic examples working?



Month 27 (with EasySurrogate now included)

How long did it take you to get each component installed (if applicable) and basic examples working?



If you have used EasyVVUQ, what was your experience and what do you think could be improved?

4 responses

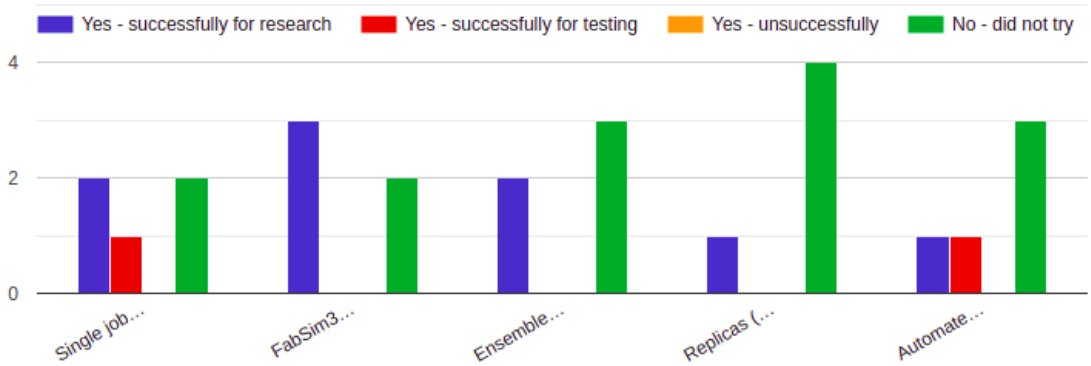
the documentation is still very poor and it is not clear what kind of analysis are available for some samplers; the name RandomSampler is a bit misleading

Good
Improvements: Is it possible to divide tutorial examples for EasyVVUQ by sampling methods? (Just suggestion)

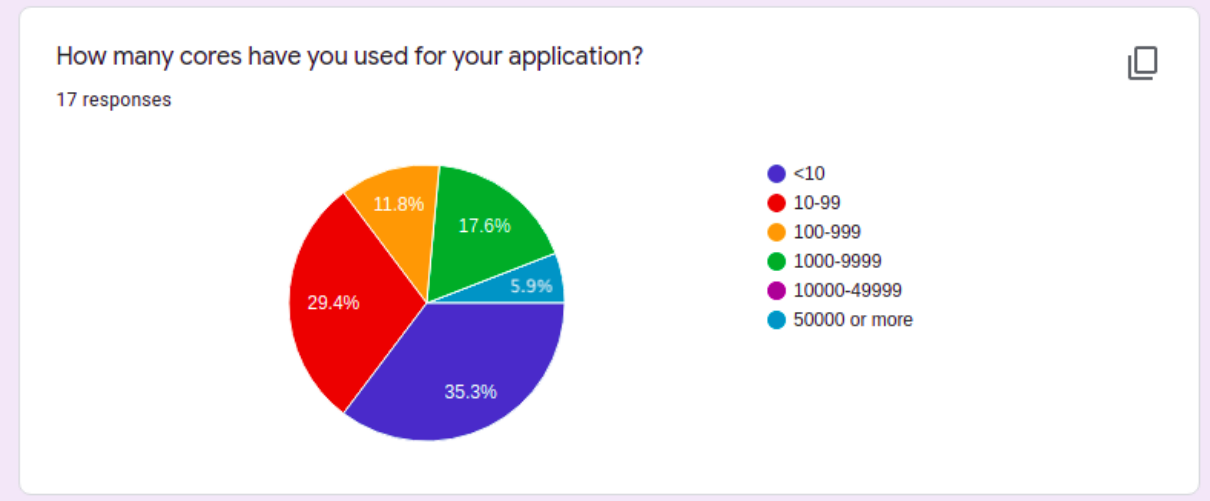
Add the option to store samples in HDF5 format. My feeling is that the database might become a bottleneck at a large number of samples

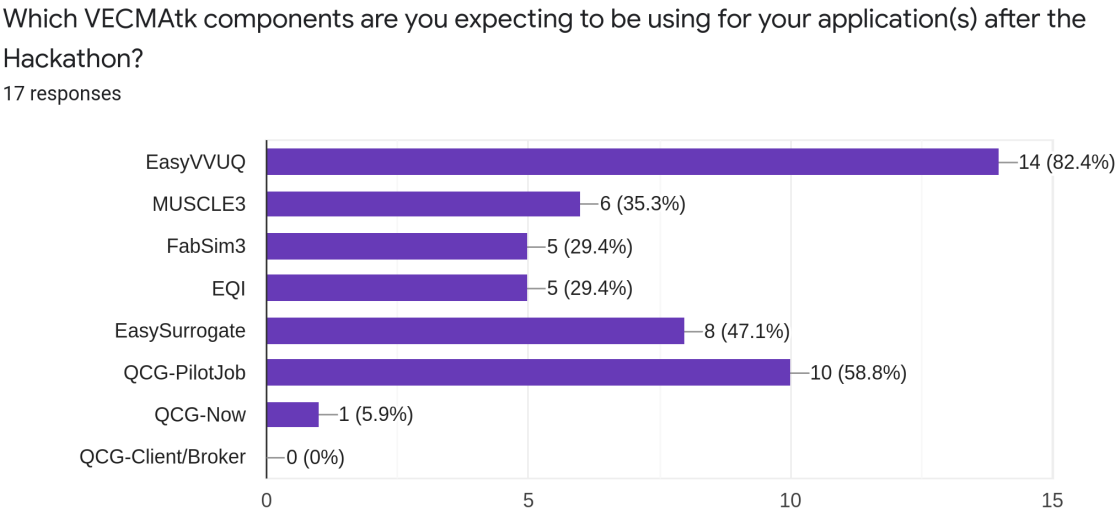
It is generally nice, and easy to use. I think it might be good to add the functions to be able to customize the name and the directory design of sample generation. The default one is nice for the UQ analysis. However if we also want to implement sensitivity analysis for example with quasi Monte Carlo method (Sobol sequence), we may want to create 4 folder (A, B, A_B and B_A) and put corresponding samples in each fold. Also, name the sample in each folder with for example A_0,A_1 and A_B_0_0, A_B_0_1. It will be much easier for computing the statistic in post-processing.

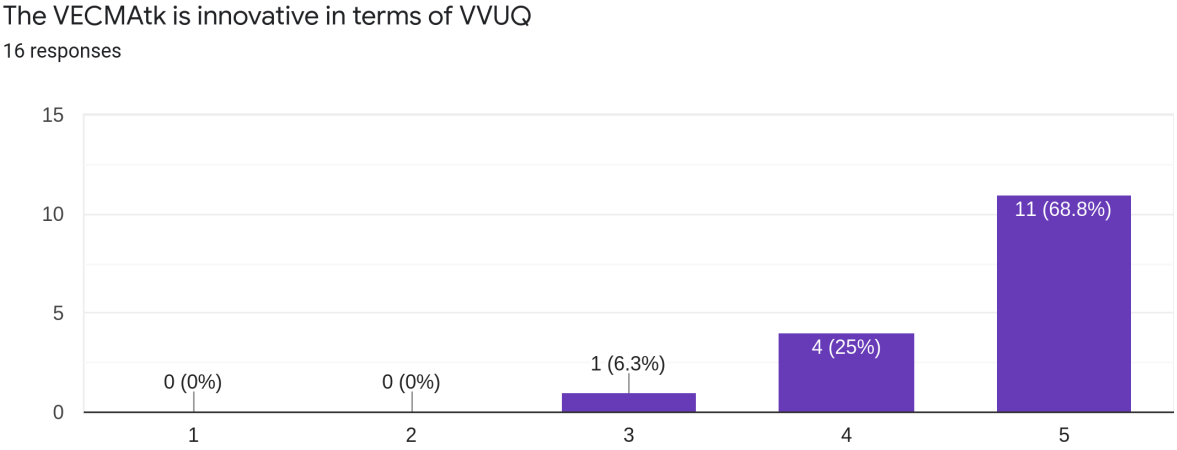
Which FabSim3 features have you tried to use?



Month 31


How many cores have you used for your application?
17 responses
11.8%
17.6%
5.9%
29.4%
35.3%
<10
10-99
100-999
1000-9999
10000-49999
50000 or more


Which VECMAtk components are you expecting to be using for your application(s) after the Hackathon?
17 responses
EasyVVUQ
MUSCLE3
FabSim3
EQI
EasySurrogate
QCG-PilotJob
QCG-Now
QCG-Client/Broker
14 (82.4%)
6 (35.3%)
5 (29.4%)
5 (29.4%)
8 (47.1%)
10 (58.8%)
1 (5.9%)
0 (0%)
0
5
10
15


The VECMAtk is innovative in terms of VVUQ
16 responses
15
10
5
0
0 (0%)
0 (0%)
1 (6.3%)
4 (25%)
11 (68.8%)
1
2
3
4
5

Month 34

Did you run into any major problems using VECMAtk? If so, please briefly describe the issue(s) here.

7 responses

- Breaking changes not detailed when upgrading version of EasyVVUQ
- No major problems, all minor issues were addressed during the hackathon (thanks!)
- No, I needed to make some changes to use Altair, but nothing insurmountable.
- N/A
- The docs/tutorials often did not reflect the current state of the code.
- My own lack of Python expertise was the main problem.
- I had to adapt code to new EasyVVUQ syntax

Month 35

What are the most important things we should add to, or improve in, the VECMA toolkit, in advance of the next major release?

6 responses

- Solid error reporting and attractive tutorials.
- Consistency between different parts of EasyVVUQ; better integration of EasyVVUQ and EasySurrogate; better integration of EasyVVUQ/EasySurrogate with QCG-PJ
- Improve the tutorials/documentation, keep the API stable
- update documentation
- Based on my colleagues' experience, fixing bugs / making sure documentation complete.
- have better documentation for each toolkit