REPORT ON OpenDreamKit DELIVERABLE D2.17
Introduce OpenDreamKit to Researchers and Teachers as laid out in Task 2.6

MIKE CROUCHER, HANS FANGOHR AND NICOLAS M. THIÉRY

Due on	31/08/2019 (M48)
Delivered on	15/10/2019
Lead	University of Leeds (ULeeds)

Progress on and finalization of this deliverable has been tracked publicly at:
`https://github.com/OpenDreamKit/OpenDreamKit/issues/252`

DELIVERABLE DESCRIPTION, AS TAKEN FROM GITHUB ISSUE #252 ON 2019-10-15

- **WP2:** Community Building, Training, Dissemination, Exploitation, and Outreach
- **Lead Institution:** University of Sheffield
- **Due:** 2019-08-30 (month 48)
- **Nature:** DEC: Websites, patent fillings, videos etc.
- **Task:** T2.6 (#29) Introduce OpenDreamKit to Researchers and Teachers
- **Proposal:** p. 38
- **Final report** (sources)

Notebooks -- interactive documents mixing prose, code, outputs, and visualisation -- have existed for decades; we may cite the Maple, Mathematica, or SageMath notebook. There is a long time experience in the community of using such notebooks for a variety of purposes in teaching and research, ranging from scratchpad for interactive computation to narratives such as interactive teaching documents or logbooks of computational research.

At the time of writing the OpenDreamKit Proposal in 2014, the Jupyter notebook was an emerging technology which had recently evolved from a generalisation of the IPython notebook to support a variety of programming languages or interactive systems. Based on modern web technologies, with a modular design informed by many former implementations, and a growing ecosystem of related tools, it showed strong potential as core user interface component for building Virtual Research Environments. Of particular interest are the Jupyter widgets that bridge the gap between interactive computation and interactive visualisation and applications, in effect making for a smooth continuous learning curve from user to power user to developer, and enlarging the range of use-cases.

In Task 2.6 (#29), we have disseminated Jupyter based Virtual Research Environment to students, researchers, and teachers to act as multipliers of knowledge and dissemination. This covered the technology but also best practices. The notebook, despite all its benefits, also has pitfalls, with which the participants have extensive experience.

As reported on in our periodic deliverables on Community building: Impact of development workshops, dissemination and training activities D2.6 (#46) , D2.11 (#36) , D2.15 (#40), OpenDreamKit participants actively organised or participated in 40 events where they advertised and delivered training on OpenDreamKit technology. In this report we start by reviewing other evaluation and dissemination activities that were carried out during the project. In a second section, we briefly reflect on the lessons learned. Indeed, that was also first and foremost the occasion to gather first hand testimony of the technology on the battle field, this to continuously inform and motivate the project activities.

Contents

Foreword

Training and dissemination is at the heart of OpenDreamKit; most participants had been active in this area for a long time before OpenDreamKit and the project was the occasion to get more resources for such activities. Sheffield (then Leeds) took the the lead until its members were hired away from academia to industry positions in Fall 2018. This did not reduce the overall dissemination activities of the project: indeed, the freed resources were redistributed to other participants that were eager to organize more activities than originally planned. There was some impact however: with continued leadership some more of the lessons learned at the occasion of those activities could have been formally collated, when currently many are in the state of shared folklore. Luckily this information is still spreading in the community through many channels: informal discussions, blog posts, mailing lists, etc.

1. Evaluation and dissemination activities

As reported on in D2.2, D2.11, and D2.15, OpenDreamKit participants actively organised or participated in 40 events where they advertised and delivered training on OpenDreamKit technology. We review in this section the other evaluation and dissemination activities that were carried out during the project.

1.1. **Teaching with Jupyter**

Many – if not most – of the OpenDreamKit participants engaged actively in using and testing Jupyter technologies in their daily teaching duties. Here are some striking examples:

- At Southampton, OpenDreamKit experts on Jupyter delivered multiple courses at the National Centre for Doctoral Training in Next Generation Computational Modelling (NGCM), which was the occasion to evaluate and disseminate OpenDreamKit technology, notably the tools developed there: D4.8: "Facilities for running notebooks as verification tests", D4.6: "Tools for collaborating on notebooks via version-control".

  This included the NGCM Summer School 2016 and 2017 at Southampton, which attracted many PhD students and some researchers from the UK, Europe and some from overseas.

- Paris-Sud participants used Jupyter in a variety of classes (data analysis, programming, computer graphics, graph theory, computational combinatorics, computational algebra, experimental mathematics, numerical analysis) throughout the math and computer science curriculum. An highlight is a 400 students class *Introductory programming in C++*. Having a uniform user interface across systems (C++, Python, SageMath, ...) was a major selling point by enabling students to be immediately productive in their new environment. The teaching was supported by the deployment of a local JupyterHub Virtual Environment and the occasion to experiment with various tools including the notebook converters nbsphinx and nbconvert, the C++ interactive interpreter xeus-cling, Jupyter widgets, interactive web pages with ThebeLab, assignement handling and grading with nbgrader. This led to an invitation of the course leader to a workshop in Edinburgh to share experience and participate in an nbgrader coding sprint where several contributions were submitted and accepted.
- Gent participant introduced SageMath and Jupyter in a variety of mathematical courses and engaged other teachers in the process.
- Since 2017, the University of Versailles uses JupyterLab as the working environment for all computational courses in the "Applied Algebra" Master's program, which enrolls around 50 students. Through it, the students have access to the Jupyter Notebook, a Unix shell (Bash), and tools to code in C, Python, SageMath, and many more languages.

  This JupyterHub based Virtual Environment deployment is described in detail on the OpenDreamKit blog[1]; the associated GitHub repository[2] has been *starred* 53 times, *forked* 38 times, and the author regularly receives requests for help, showing that there is a growing interest in deploying similar setups at other universities and research centers.

  The VRE has been opened in 2019 to the whole university, with an expanded set of tools. In the upcoming academic years other programs will start using it for their courses.
- The University of Silesia uses elements of OpenDreamKit in many aspects of education. JupyterHub is installed on the campus facilities and it is available for students. Currently there are ca. 1000 users. There is an experimental setup for research backed by SLURM HPC cluster and jupyter-batchspawner. There are courses for physics and biophysics students which are using interactive books produced during OpenDreamKit. Data science oriented courses (Introduction to AI and Machine Learning) are fully based on JUPYTER and nbgrader system for distribution and collection and grading of excercises. Courses in PYTHON programming for computer science students are using JUPYTER notebook. There are experimental attempts to use IJava kernel with JUPYTER for introductory Java course. SAGEMATH is presented to students of physics and biophysics during first course in ICT and then used during whole course.
- In USTAN, Alexander Konovalov introduced Jupyter notebooks for teaching Python in the 2nd year module. Lecture materials were provided in the form of Jupyter notebooks, which both lecturer and students were able to execute during the lecture. One of the course practicals consisted of producing a reproducible report about analysing a dataset, which had to be submitted in the form of a Jupyter notebook. Furthermore, Konovalov demonstrated the usage of Jupyter notebooks for sharing reproducible computational experiments using Microsoft Azure and Binder in a number of workshops for USTAN staff and research students.
- Sheffield OpenDreamKit participants spearheaded a transformation in the way computation was taught at The University of Sheffield across multiple subjects and at many levels. The Department of Physics, for example, now teaches *all* of its undergraduates how to program in Python using Jupyter notebooks and the CoCalc (formerly SageMathCloud)

[1] `https://opendreamkit.org/2018/10/17/jupyterhub-docker/`
[2] `https://github.com/defeo/jupyterhub-docker`

cloud-based environment. Furthermore, many other physics modules at Sheffield now include some type of computation using the same technologies. Programming is no longer considered a separate topic but the integral part of modern science that it really is.

Working with an Italian Marie-Curie fellow in Bioinformatics, Sheffield's OpenDreamKit participants developed a set of short postgraduate 'Bioinformatics Awareness Days' which used OpenDreamKit technology to introduce Bioinformatics workflows to clinicians at Sheffield Institute for Translational Neuroscience. This further led to an OpenDreamKit teaching tutorial being held at University of Naples Parthenope.

Other subject areas where Sheffield's OpenDreamKit participants assisted in developing enhanced lecture material using OpenDreamKit technologies include Machine Learning, Mathematics, Computer Science, Biology and High Performance Computing.

1.2. Teaching material

Interactive lecture notes are an area where commercial vendors such as MapleSoft and Wolfram Research are spending a lot of time and money developing material. Within the Jupyter ecosystem it has become possible to author interactive lecture notes and make them openly available (e.g. through BinderHub). We have created such interactive lecture materials, used them in university education, and made them openly available on the Internet. This includes four interactive textbooks and templates for authoring such books (see D2.9: "Demonstrator: interactive books on Linear Algebra and Nonlinear Processes in Biology" and D2.14: "Demonstrators: Problems in Physics with Sage, Computational Mathematics for Engineering"), but also Software Carpentry lessons, course notes, course templates, etc.

Anecdotal evidence and feedback from individual users (see D2.14) shows that they are used outside the OpenDreamKit partners, in and out of Europe, both by individual students and university lectures.

1.3. Local consulting

Across all the OpenDreamKit partner sites, OpenDreamKit staff and PIs have engaged with colleagues, students and decision makers to advocate the benefits of the Jupyter research environment; often effectively serving as consultants for best practice computational mathematics and science tools and workflows.

For example, at Sheffield, a taster seminar (1-2 hours) and follow-up short course (1-2 days) on Jupyter for lecturers and researchers were organised targeting and engaging science and engineering disciplines beyond mathematics as the Jupyter ecosystem of tools is of value for teaching and research in any discipline having to work with computational and data based research. These workshops were integrated into the research support services of the Sheffield IT department, and integrated with the Research Software Engineering movement that originated in the UK over the previous years.

The OpenDreamKit funding was essential in forming Sheffield's Research Software Engineering (RSE) Group, one of the first such groups in the UK. Providing training, documentation and support in the use of OpenDreamKit technologies in both teaching and research helped the Sheffield RSE group demonstrate to the University how vital RSE support can be. This directly led to the fully-funded, diverse group that now exists at Sheffield which has served as a model for many other such groups around the UK, Europe and more, recently, the United States. This was documented in our blog post *How OpenDreamKit supported the RSE revolution*.

2. Evaluation and adoption

By design, the target audience for this project is extremely diverse, and the toolkit approach makes it so that there isn't such a thing as a well defined end-product of OpenDreamKit. Hence trying to capture the adequateness and the adoption through, e.g., polls or metrics makes limited sense. Instead, we focused on gathering feedback through first hand experience, and continuous contacts with a large variety of end-users during all the activities reported on above.

To give a sense of the lessons learned, we reflect in this section on the adequacy for users and adoption by the community of some of the tools we have promoted.

2.1. **Computational systems**

OpenDreamKit's computational systems are large pieces of software (1.5M lines of code for SageMath), which take time to master. Nevertheless the learning curves have considerably improved. A major step has been the ease of access for complete beginners: in earlier years, most of the first day or two was spent struggling with software installation. Now installation is much easier, and can be deferred as desired by starting online. Beginners are thus immediately productive for simple tasks.

Meanwhile the collection of training material has grown considerably, many of them being available as Jupyter notebooks help a lot for the autonomy of the students especially in the early steps. This includes Software Carpentry style lessons as well as the open introductory book "*Computational Mathematics with SageMath*", whose translation from an earlier French version was partially carried out under ODK funding.

In practice most of our tutorial sessions can now be self-guided, with participants exploring the material following their pace and interest. The instructors can focus on a few presentations giving the broad eye view (ecosystem, underlying concepts, best practices, ...) and on individual help during tutorials (personalized tutorial suggestions, debugging help).

As anecdotal evidence of the interest of the mathematical community, let us mention that the aforementioned book is downloaded 500 times every day; we also point to the strong attendance at our many dissemination events; of particular interest are the informal polls run during the FPSAC conference which show that, in this area of mathematics, about one half of the researchers use computer exploration on a regular basis, while a large fraction of them often writes code for their research.

2.2. **The Jupyter Notebook**

Based on our dissemination activities, the Jupyter Notebook is emerging as one of the major user interfaces for computational (pure) mathematics, if not the de-facto standard in many communities. A key selling point is having a uniform interface across computational tools, thanks notably to D4.7: "Full featured JUPYTER interface for GAP, PARI/GP, Singular".

This is in line with the general evolution in areas such as scientific computing and data sciences. As anecdotal evidence, the use of notebooks at Sheffield, Southampton, XFEL and other universities has grown significantly. Also, the integration of Sun Grid Engine and Project Jupyter, done as part of OpenDreamKit (D5.3), has led to educators using the notebook to introduce various aspects of High Performance Computing.

We now detail two typical lessons we have learned.

Practice over hundreds of students in class or other learners at dissemination workshops have shown that it takes no more than an hour of guided instruction to become sufficiently acquainted with the basics of the Jupyter Notebook to be able to autonomously explore properly structured collections of notebooks (e.g. an interactive text-book, or a collection of tutorials in a workshop). At this stage, the simple, linear narrative structure of a notebook is a precious guide to the reader. It does take some practice however for the user to not get any more confused by the potential disprecancy between the visual order and execution order of cells. This can be mitigated by short notebooks, and the use of notebook extensions that enforce the execution order of cells.

Properly authoring one's own notebooks also takes a lot of practice. The simplicity of incrementally building code by testing code snippets and aggregating them is a big strength of the notebook. It's also a weakness: notebooks have a tendency to grow organically and become bloated and unstructured. This natural tendency must be explicitly counteracted through training and the sharing of best practices.

2.3. **Interactive widgets with Jupyter**

Among the technologies we disseminate, Jupyter Widgets are one with the largest "wow" factor, sparking ideas among our users on how they could be exploited for, e.g., teaching or research. As we anticipated, the learning curve remains steep however. Most users stay at the basic stage of authoring so-called interacts. Few actually take advantage of the full flexibility offered by widgets to build interactive applications by combination thereof. Authoring one's own completely new widget (e.g. for a new type of visualization) takes specific expertise, notably to implement the JavaScript side of it. One strategy to tackle this is to have Research Software Engineers leverage the technology to non experts by implementing generic widgets that can be specialized to cover a range of use cases. This approach was validated at the occasion of two of our dissemination events at CIRM *Free Computational Mathematics* and Ljubljana *SageDays 105* where the generic widgets of D4.16: "Exploratory support for semantic-aware interactive widgets providing views on objects represented and or in databases" were presented, and then adopted and extended by several PhD students. See Figure 1 for an example.

The same adoption pattern was witnessed for our 3D visualization widget D4.12: "JUPYTER extension for 3D visualisation, demonstrated with computational fluid dynamics".

2.4. **Notebook validation and version control with nbval and nbdime**

Our development and dissemination activities on NBVAL and NBDIME (D4.8, D4.6) within OpenDreamKit has resulted in researchers switching to using the notebook for all of their software documentation and tutorials. They also confirmed in practice the unique selling point of NBVAL: ensuring that notebook-based documents stay up to date and functional even as the underlying software stack and environment evolves. In particular, NBVAL allows notebook-based documentation to become part of the formal testing and continuous integration framework.

2.5. **Interactive documents**

We refer here to Section 4 of D2.14: "Demonstrators: Problems in Physics with Sage, Computational Mathematics for Engineering".

2.6. **Class management**

There are various methods for managing classes using OpenDreamKit technologies and OpenDreamKit participants have tried most of them. None of them are perfect and many non-specialist lecturers require support in choosing and using the various options which include:

- Commercial cloud-based environments such as Microsoft Azure, Google's Colab and CoCalc. These are very easy to set up and use. The main caveat is the loss of control, notably in term of private data protection. Practical issues included updates occurring in the middle of exam sessions that modified user-interface behaviour and even computational results in some cases. For one computationally intense course, the amount of CPU power available in the cloud environment was insufficient for students to complete some project work which led to complications.
- Bring your own laptop. Such sessions ensure that participants leave teaching sessions with a fully functioning computational environment but supporting diverse operating systems and hardware can be extremely challenging for those running the course.
- Server or computer lab run by University IT. They can provide a very controlled and stable environment but at the cost of teachers not being able to update in a timely fashion unless the servers are supported by dedicated Research Software Engineering staff.

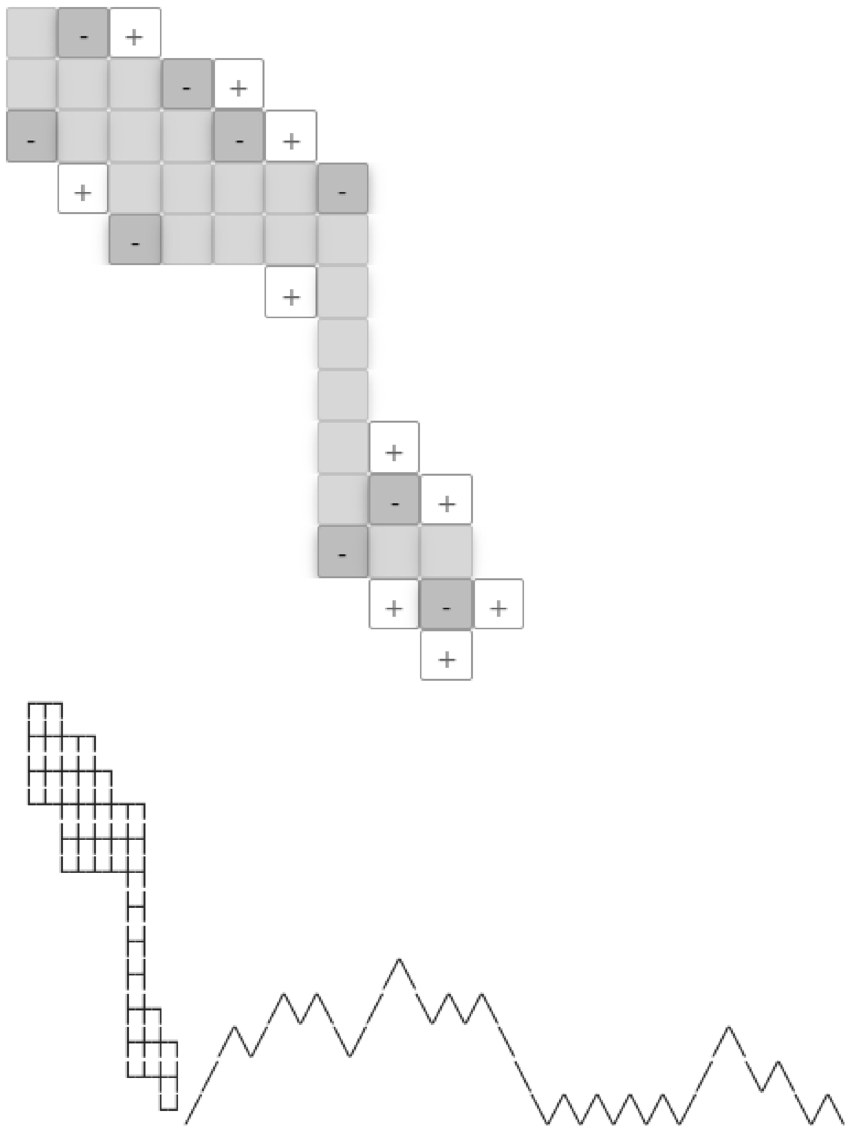

FIGURE 1. An interactive widget to manipulate *parallelogram polyominos* (*source notebook*). Clicking on the corners of the polyomino in the upper figure adds and removes them. The two mathematical representations of the same object below (in unicode and as a Dyck word) are updated in real time. This may not be so impressive in itself. The point is that, based on our work, it was produced in a matter of hours by a PhD student; furthermore, a researcher or teacher can quickly adapt the input widget for similar objects or interactions, and use it as building block for her own applications.

- Server or software environment in the computer lab setup by the teacher. This is the most flexible solution, but requires strong computer literacy, and also the agreement of the University IT. The main difficulty is the integration with the local information system.

When using this type of technology across many subjects in a university, it quickly becomes apparent that, at this stage, the only way to fully support lecturers properly is to have on-site specialists that can provide the required assistance. Such work forms the foundation of the Research Software Engineering groups that have started to form all over the world. OpenDreamKit participants formed the first demonstrations of such partnerships.

On the technical side, one of the most popular tools for managing assignments in a class is nbgrader; it supports authoring, production and distribution of instructor and student version, collection, semi-automatic grading, feedback. It is modular and each piece can be used either from a graphical user interface (UI) within Jupyter or from the commandline. This gives, in principle, quite some flexibility to integrate it into the local information system and workflow.

Experience shows that usage through the UI is straightforward, for students and instructors alike, even novices. However, as of now, setting it up requires good computer literacy, if not admin rights on the local system. The tricky part is to setup the *exchange zone* through which assignments are exchanged back and forth between instructor and students. Support for multiple courses and multi-instructor courses is also very recent. We expect the entry barrier to decrease greatly with the ongoing development of an *exchange service*; once deployed locally at an institution and integrated with the local information system, instructors will be able to seamlessly setup new courses.

2.7. **Packaging and distribution**

See D3.10: “Packaging components and user-contributed code for major Linux distributions”.

2.8. **Live publication with Binder**

Thanks to public forges like GitHub or GitLab, users can easily publish their notebooks online, and readers can then download them or browse them online. Installing the required software to actually run the notebook remains a burden for the reader. Binder is a Jupyter-based web service that bridges that gap. Provided that the repository holding the notebook is *Binder-enabled*, that is the the author has added annotations specifying the software requirements, readers just have to open a URL in their browser to run the notebook in a temporary virtual environment. Hence achieving immediate reproducibility (long term reproducibility brings lots of additional complexity).

For readers it can’t be more easy. In fact, Binder URLs have become our tool of choice to kickstart learners at the very beginning of our training workshops, typically by pointing them to `opendreamkit.org/try`. Thanks to this simplicity for readers, Binder is used extensively, constantly serving hundreds of temporary virtual machines concurrently.

For authors, the process requires Basic Lab Skills (as taught in e.g. Software Carpentry events) and copying from a template. This is straightforward but remains an entry barrier. Our experience is that it takes walking the author through the process once (≪see, it’s easy≫), and then it becomes a regular practice.

The first Binder-enabled notebooks for SageMath were published about two years ago. Nowadays, out of ten thousand SageMath notebooks published on GitHub[3], about one hundred are configured for Binder[4]. About half of the authors were directly trained by us. We expect the uptake to grow widely in the coming years, as the practice spreads by word of mouth.

Of course the numbers are proportionally much higher in larger communities: there are millions of Jupyter notebooks on GitHub and, from search for a configuration file `environment.yml`, one may estimate that tens of thousands are Binder-enabled. In particular a large fraction of library developers for, e.g. Python, provide online demonstrations using Binder-enabled notebooks.

2.9. **Integration with EOSC**

All technologies developed in OpenDreamKit are based on open, standard, and modern web technologies, which should make it straightforward to integrate in any larger infrastructure or

[3]`https://github.com/search?q=SageMath+extension:.ipynb&type=Code&ref=advsearch&l=&l=`

[4]`https://github.com/search?q=SageMath+path:/+filename:Dockerfile&type=Code&ref=advsearch&l=&l=`

infrastructure federation, and notably the European Open Science Cloud. This claim is well supported by two facts: first, a JupyterHub service was deployed in 2017 by EGI as a service for the EOSC, and has been running continuously since then; EGI is eager to participate in the consortium of spin-off projects of OpenDreamKit. More generally, a number of projects working towards the European Open Science Cloud (EOSC) have chosen the Jupyter ecosystem as technology to realise the EOSC – in particular for remote access, remote analysis and visualisation of data, and for improved reproducibility.