REPORT ON OpenDreamKit DELIVERABLE D1.7

Innovation Management Plan v2

NICOLAS M. THIÉRY ET AL.

Due on	31/05/2019 (M45)
Delivered on	04/09/2019
Lead	Université Paris-Sud (UPSud)
Progress on and finalization of this deliverable has been tracked publicly at: `https://github.com/OpenDreamKit/OpenDreamKit/issues/23`	

DELIVERABLE DESCRIPTION, AS TAKEN FROM GITHUB ISSUE #23 ON 2019-09-05

- **WP1:** Project Management
- **Lead Institution:** Université Paris-Sud
- **Due:** 2019-05-31 (month 45)
- **Nature:** Report
- **Task:** T1.3 (#16) Innovation Management
- **Proposal:** p. 33
- **Final report** (sources)

In the first version of the Innovation Management Plan for OpenDreamKit D1.4 (#20), we reviewed some context about the Virtual Research Environment and the end-user / developer relation, described transversal innovations OpenDreamKit is pushing for, provided notes on open source (mathematical) software development processes, with a focus on some systems the VRE is developing, and presented elements of strategy for reaching out for a wide range of end-users.

This second version complements this with two sections. In the first one, we review the choice and impact of open licenses in the context of OpenDreamKit, exemplifying potential issues with concrete examples where the project was actually impacted. Indeed, if most of the usual Intellectual Property issues were easily taken care of in this project thanks to the systematic use of open licenses, a few caveats exist nevertheless. In the second section, we review the different types of outcome of the project and discuss their respective sustainability.

This second version also includes minor updates to the earlier sections, notably to reflect the work plan revisions that occurred after Reporting Period 1.

CONTENTS

1. CONTEXT

OpenDreamKit (Open Digital Research Environment Toolkit, ODK for short) is a four-year Horizon 2020 European Research Infrastructure project (#676541). In the 2015-2019 period, it provided substantial funding to the open-source computational mathematics ecosystem, and in particular popular tools such as LINBOX, MPIR, SAGE, GAP, PARI/GP, LMFDB, SINGULAR, MATHHUB, and the JUPYTER interactive computing environment.

From this ecosystem, ODK has delivered a flexible toolkit enabling research groups to set up Virtual Research Environments (VRE), customised to meet the varied needs of research projects in pure mathematics and applications, and supporting the full research life-cycle from exploration, through proof and publication, to archival and sharing of data and code.

The primary end-users are researchers in (pure) mathematics; however, thanks to the modular design the work will benefit a wide range of end-users, including teachers, engineers, in academia, public institutions, or enterprises.

An unusual feature of ODK is that it emerged from an ecosystem of community-developed software where the prevalent development model is "by users for users". And indeed, many of the ODK participants are themselves end-users: academics wearing several hats – developer, researcher, teacher, in mathematics or computer science, etc. – who gathered to develop the computational tools they needed for their daily research. ODK itself was born from two dual movements, developers pushing innovation and end-users pulling innovation, often led again by people wearing both hats. This, combined with long dissemination experience from said people, is a strong asset for delivering products which actually benefit a wide range of end-users.

Reciprocally, ODK works hand-in-hand with relevant communities; some innovations brought to the toolkit are partially or entirely accomplished outside of the ODK project. This is not a problem in itself as the communities and the project share the same values and objectives.

End-users will benefit from innovations brought by ODK in the multiple pieces of software forming the open-source VRE. In the case of ODK the innovation will be the unification of open-source tools with overlapping functionality, the simplification of the tools for end-users without coding expertise, and the development of user-friendly interfaces.

The following document reviews the innovations that ODK brings to end-users, then explains the processes enabling the implementation of said innovations, and how ODK targets its end-users. Most of the usual Intellectual Property issues were easily taken care of in this project thanks to the systematic use of open licenses; a few caveats exist nevertheless and we briefly discuss the choice of licenses, exemplifying potential issues with concrete examples where the project was impacted. We conclude by reviewing the different types of outcome of the project and discuss their respective sustainability.

2. INNOVATIONS OPENDREAMKIT BRINGS TO THE RESEARCH COMMUNITY

ODK has several cross-cutting objectives, each of which brings innovation to the research community:

- Develop and standardise math software and data for VRE: WP3, WP4, WP5, WP6
- Develop core VRE components: WP3, WP4, WP5, WP6
- Bring together communities: WP2, WP3
- Update a range of software: WP3, WP5
- Foster a sustainable ecosystem: WP3, WP4, WP5, WP6
- Identify and extend ontologies: WP6
- Effectiveness of the VRE: WP2
- Effective dissemination: WP2

We explore two typical areas of innovation in more detail:

Best practices and tools for correct and reproducible research: In the excellent talk "Is your research software correct", Mike Croucher highlights crucial best practices for the use of software in research, including open code and data sharing, automation, use of high level languages, software training, version control, pair programming, literate computing, and thorough testing. A lot of work in ODK relates to disseminating this set of best practices (WP2: "Community Building, Training, Dissemination, Exploitation, and Outreach"), and enabling it through appropriate technology (WP4: "User Interfaces"). Just to cite a few examples, D4.6: "Tools for collaborating on notebooks via version-control", and D4.8: "Facilities for running notebooks as verification tests" enable respectively version control and testing in the JUPYTER literate computing technology. Mike's talk has been delivered in several of ODK's many training events (see especially D2.2: "Community building: Impact of development workshops, dissemination and training activities, year 1" and D2.11: "Community building: Impact of development workshops, dissemination and training activities, year 2 and 3").

Multisystem architecture: Modern research increasingly requires the combination of multiple computational, database, and user interface components. We explore novel ways to combine software while taking advantage of parallel features, sharing data and semantics in a sound way, and fostering collaboration between systems. For example, a user can use the Jupyter interface to write in a high-level programming language like SAGE (see D4.5) using low-libraries like PARI (see **T4.12**) running high-performance computations (see D5.16). While doing this, they can access the Cremona database of elliptic curves, a small part of the LMFDB (see **T6.8**). Since all this is happening in Jupyter, the user can easily build in interactivity and visualisations (see WP4).

Uniform front-end to computational research results: The Jupyter project, and especially (but not limited to) the Jupyter Notebook interface (formerly the IPython Notebook) were conceived of in large part due to the recognition that multiple high-level, general-purpose programming languages in computational research (Julia, Python, R) all had shared need for a notebook-like format for sharing research, and that the existing architecture was more-or-less applicable to all three, and potentially to other programming languages, and more specialized applications. We have proven that, indeed, the Jupyter notebook interface/metaphor can be extended to the open mathematics research community, with notebook interfaces for SAGE (which is already Python-based and had its own notebook format prior to Jupyter), as well as more specialized CAS's with their own, relatively niche programming languages such as GAP, PARI, and SINGULAR. We have also shown the benefit of being able to reuse the same file format (the `ipynb` format) to share tutorials and computational research results across all these different systems, especially in conjunction with emerging technologies such as Binder, which allow execution of these notebooks on the cloud, with zero installation for end-users.

Open collaborative project preparation and management: The OpenDreamKit participants are long time Open Science enthusiasts. Accordingly, following paths that are now finally well established: all the project outcomes are Open Source, Open Data, and Open Publications. OpenDreamKit introduced one additional piece of innovation by exploring Open Project Management in the context of large European Projects: all the steps of the project — preparation, production, management, reporting, dissemination — were carried out in the open and in strong collaboration with the community.

The outcome is overwhelmingly positive; we have actively promoted this approach, e.g. in the context of several blogs and presentations, and we hope that it will be widely adopted.

3. Implementation processes of the innovations

The organization of the innovation processes plays a key role in the success of the software in the ecosystem ODK builds upon. Their communities have, over the years if not decades, developed and accumulated a strong expertise in the social engineering aspects of community software development, pulling general ideas from the open- source movement (e.g. public development, early releases, ...), and adapting them to their specific contexts. They are heavy users of the usual collaborative software development tools and best practices such as mailing lists, wikis, collaborative editing pads, online chat rooms, version control, issue trackers, continuous integration, regression testing, code reviews, coding sprints, etc.

We describe below some striking aspects of the implementation process in some of the software systems involved in ODK.

3.1. **Implementation process of SAGE**

All the development happens in the open (public mailing lists, bug tracker, ...). There is no specific sustainable leader in the development of SAGE. Its "community" of developers bases its work on the consensus of the group and on the availability of its members to tackle issues and work on the software development. If a decision doesn't create a consensus, there can be a vote but that seldom happens. When any kind of change (new feature or bugfix) is proposed to the software, it is reviewed by at least one other member of the community, more often than not with others looking over the shoulder.

Regarding the development process of SAGE, arguably, OpenDreamKit accelerated trends that had just started to take a foothold before the beginning of the project. Historically, all changes and additions to SAGE source code were discussed on the issue tracker hosted at `https://trac.sagemath.org`, and were only accepted after review by a member of the community. While this process guaranteed the quality of the code, it also hindered agility, a complaint often heard on SAGE forums. Indeed, while it is possible, with good planning and management, to create and evolve large thematic subprojects inside SAGE (a recent example being the Coding Theory framework), the review process still encouraged small incremental changes over large code removals/additions.

This workflow has nowadays evolved in two ways: first, with the externalization of several core components to separate projects (e.g., UI migrated to Jupyter, PARI/GP interface migrated to CyPari, signal handling delegated to Cysignals, ...), the fine-grained code review process for these now happens outside of SAGE, in the respective issue trackers, using whatever processes and tools those projects may see fit. At the level of SAGE, the quality of the included code is still guaranteed by the standard review process at the moment of integrating the component, however this review is much coarser, and concerned more with integration than code quality. The alignment between a project's decisions and SAGE's own goals is still guaranteed by having the SAGE community participate, or even pilot, the development process of the component.

Second, new SAGE contributors who wanted to share their code were historically encouraged to include it in the SAGE sources. While this guaranteed maximum availability of the code, it also lead to several cases of *bit-rotting* inside the SAGE codebase: old code whose developer had left the community and that no one was interested in keeping up to date. More and more, especially at OpenDreamKit dissemination events, new users are encouraged to learn about the basics of packaging and distributing Python code through public repositories (PyPI in particular). The facilities added to the SAGE executable let users easily install packages from public repositories, thus making code sharing easy and natural in SAGE. This way, end users maintain control over their SAGE projects; if one of these projects gains enough popularity to be wanted as a basic component of SAGE, the community can take over maintenance and integrate it to the main code base. This model makes it easier to maintain a relatively stable and high-quality *core* library of basic functionality in SAGE, while giving the wider community more agility to

develop easily shareable code which *uses* SAGE without being hamstrung by the more stringent maintenance requirements of the core library. This is similar to the model demonstrated by the Python *standard library* itself, which places high requirements on the inclusion of major new functionality, while preferring to emphasize development of third-party libraries outside the standard library.

Experience has shown that the success of SAGE, and in particular the biggest achievements and best innovations, owe much to a long track record of focused week-long workshops, called Sage Days (about ten per year since 2005), where developers get together and form small groups for focused coding sprints. This method is actually close to the "two pizza" team rule from Jeff Bezos, the founder and CEO of Amazon. According to Mr. Bezos,"If you can't feed a team with two pizzas, it's too large", beyond 6–7 people, the more you add people to a group, the less the group is agile and innovative because too much effort is put into communication and management.

An example of such workshop is Sage Days 77, organised by Nicolas Thiéry in April 4–8 2016, in a guest house far from urban civilization and with a solid internet connection. About 15 people joined this workshop throughout the week, and split into three or four constantly self-reorganizing teams. Concerning the impact, proper packaging and distribution has been a recurrent issue for SAGE and is a major task for OpenDreamKit (**T3.3**: "Modularisation and packaging"). Major brainstorms occurred during the week to clarify the needs, isolate the core difficulties, and explore potential approaches to tackle them. The outcome was posted on the Sage Wiki, to be shared and further edited by the community. This fostered tighter collaboration between the packaging efforts for various Linux distributions, and triggered major progress on the Debian packaging side. Several small workshops such as this one are to be organised all project long to speed up the software development process.

3.2. Implementation process of JUPYTER

The JUPYTER project is driven by the Jupyter steering council, as described in the Jupyter governance documents. This council is composed of 15 members, one being Benjamin Ragan-Kelley who is leader of the Work Package 4 (User Interfaces), at least two being regular participants to ODK events.

The larger-scale mission of the project is decided by the steering council. In most cases, the wider JUPYTER community decides what should be done by contributing proposals on an individual basis. Most proposals come in the form of pull requests on GitHub, but larger proposals can be discussed as enhancement proposals beforehand, and must be approved by the steering council. In general, decisions for additions are handled by the maintainers of existing packages who are longstanding members of the community or delegates thereof (such as those hired under the ODK grant). If there is conflict, decisions can be resolved by the steering council.

Concerning the developments reviews, the JUPYTER community does it publicly. The maintainers of each project take on the bulk of the review responsibility. If there are conflicts, they can be brought to the steering council for resolution.

3.3. Implementation process of SINGULAR

Wolfram Decker and Hans Schönemann from the University of Kaiserslautern are considered to be the leaders of the SINGULAR software. There is no specific process to decide how the software should evolve. It can be individual decisions of a single person in need of a service for their research, or decision taken by vote by the core of SINGULAR developers during a meeting. Nevertheless, essentially all new code is reviewed by Hans Schönemann.

3.4. Implementation process of GAP

Individuals who have helped or been helping in the development, the maintenance, the advisory and user support roles are referred as the GAP Group. Furthermore, the GAP Council was

formed in 1995, which currently consists of 19 senior mathematicians and computer scientists with Prof. Leonard Soicher as chair. The GAP Council is not a representative body but more resembles an editorial board, in which the Council chairman acts as editor in chief, and the other members as ”associate editors,” after the fashion of some journals.

In GAP, most issues concerning development are decided after discussion among the developers or the support team. Most of the discussions happen in GitHub issues and pull requests, primarily in the main development repository. Larger proposals can be discussed in the Open GAP development mailing list. Sometimes discussions may fail to reach a conclusion, either because the issues are too complex or because people simply can't agree. In this case a smaller core group will discuss and make decisions which have to be in the general interests of GAP.

For GAP packages, the decisions are taken by their authors and maintainers. Many of the currently 145 packages redistributed with GAP adopt an open development model which facilitates interaction between them and participation of the other GAP developers in technical discussions. For the core GAP system, reviewing and evaluation is mostly done via public code review on GitHub and regression tests. For GAP packages redistributed with GAP, there are regression tests and checklists to use when a new package is submitted for the redistribution with GAP.

Finally, GAP features a longstanding formal package refereeing process overseen by the GAP Council. This process, rather unique and exemplary among academic software, aims at promoting high quality contributions by rewarding authors with credit similar to that of a published paper.

4. The end-users targeted at by innovations

As it was previously said, the originality of the ODK VRE is its ”by users for users” model. Indeed, ODK's participants have many hats: they are mathematicians, computer scientists, developers, researchers, professors, end-users and sometimes all of this at the same time.

But since ODK is promoting open-source, it is naturally aiming at reaching out to as many end-users as possible. In order to do that ODK comprises two boards which can help reach end-users outside the individual components' developer communities and overall make the ODK work as effective as possible.

4.1. **Boards within ODK**

4.1.1. *Quality-Review Board.* The Quality Review Board (QRB) is composed of four members: Hans Fangohr (chair), Alexander Konovalov, Konrad Hinsen and Mike Croucher. They all have a long track of developing open-source research software and disseminating to end-users. Some of their objectives are to:

- Assess the quality of the software engineering aspects
- Identify best practices
- Improve future work within ODK and disseminate knowledge to a wider audience, i.e. potential end-users.

Thus, the work of the QRB can facilitate access of the end-user to the innovations brought by ODK by making sure best practices are respected and the software development is of high quality, and as sustainable as possible by following best practice software engineering. However the board specifically aimed at the end-user's needs is the Advisory Board (AB).

4.1.2. *The Advisory Board and its end-user group.* The Advisory Board (AB) is composed of seven members (Lorena Barba, Jacques Carette, Istvan Csabai, Françoise Genova, Konrad Hinsen, William Stein and Paul Zimmermann), and industry/end-users and academic representatives who understand broad 21st century needs for computational mathematics. The AB is to give an independent opinion on scientific and innovation matters, in order to guarantee:

- Quality implementation of the project,
- Efficient innovation management,
- Project sustainability.

Additionally, the Board includes a small End-user Group (3 to 4 persons out of 7) which will be connected to an informal community of end-users. They will control the project execution from the point of view of the end-user needs and requirements, making sure that the outcome of the project indeed matches those needs.

4.2. **End-users targeted by ODK innovations**

Matching user needs is not sufficient. One additional challenge is to promote the VRE to potential end-users so that it actually gets put to use, in Europe and beyond, in established developer communities and across new users, in math and other relevant research fields. This challenge is being tackled by the Work Package 2 "Community building, training, dissemination, exploitation and outreach".

4.2.1. *A worldwide promotion.* As it was previously said, ODK developed and promoted software that are open-source. The universal and cost-free distribution nature of open-source software allowed ODK to target every country and continent notwithstanding any level of infrastructure development, economic performance or lack of a solid institutional academic network. ♠**TO DO**: *Izabella: update this paragraph for the following reporting periods*♠ In the first year of the project from Sept. 2015 to Sept 2016, 14 events (workshops, schools, etc.) were (co-)organised by ODK. Some of them were of course planned in Europe and North America, but others were planned in Africa, South America and Middle East. In the next years ODK will continue at the same pace to (co-)organise events in the same geographical regions. However in some areas where the internet connection does not allow for massive cloud usage, one must find alternative solutions such as distributing the required install files using USB sticks rather than online repositories. This simple trick enables dozens of undergraduates, master students, PhD students, post-doctorates, teachers and professors following the same given school to start working on a SageMath or Jupyter tutorial at the same time.

Because of the open-source nature of the ODK software, everything can infinitely be replicated and shared without any constraint. Therefore in addition and following the events organised by the project, ODK is counting on a snowball effect for the reaching out of end-users.

4.2.2. *Established communities.* Since ODK participants are all part of at least one of the well established open-source software communities, the communication on ODK's achievements comes naturally and easily. Furthermore, many workshops are organised all year-long, whereas they are (co)financed by the project or not, during which the developer communities join together their efforts to improve the software from the ODK toolkit.

4.2.3. *Research fellows outside of established communities.* As described in D2.2: "Community building: Impact of development workshops, dissemination and training activities, year 1" D2.6: "Community building: Impact of development workshops, dissemination and training activities, year 2" D2.11: "Community building: Impact of development workshops, dissemination and training activities, year 2 and 3", ODK was introduced at many events – organized by ODK or not – in order to reach out to potential end-users outside of the regular open-source software developper communities. This included two major dissemination conferences with about 60 participants; the first one, Computational Mathematics with Jupyter, was jointly organised with the CCP (Collaborative Computational Project) CoDiMa at the ICMS (International Center for Mathematical Sciences) in Edinburgh in January 2017. The second one – Free Computational Mathematics – was organized at the CIRM (Centre International de Recherche Mathématique) in Luminy in February 2019. Both featured keynote talks from leaders of the various communities and many tutorials.

These events were the occasion to advertise OpenDreamKit's components, promote best practices, and get first hand feedback from end-users. This was also the occasion to witness the strong impact that OpenDreamKit contributions like the port of SAGE to Windows had to reduce the entry barrier for new end-users.

5. ON THE CHOICE AND IMPACT OF OPEN LICENSES

Generally speaking, IP management in the context of community developed mathematical systems causes little difficulty, thanks to a quite widely spread consensus on open science – backed up both by ethical and practical reasons – minimal economical or strategical issues, and good will from the stakeholders. Nevertheless IP issues occasionally arise while integrating complex systems, and they can become a practical burden. In this section, we illustrate this with two typical situations that occurred recently in our communities.

5.1. **License changes in decades-old systems**

The management of intellectual property (IP) developed over several decades by a distributed community of researchers with little or no input from IP professionals presents some unique challenges. In the early years of the GAP system, for instance, no reliable record was kept of the authorship of all the components of the system or of the employment status of the authors, and no transfers of IP rights were obtained. This means that many individuals and institutions own the IP of small fragments of the system, in many cases unwittingly. This does not create a problem for continuing distribution under the current license (since any contributor grants permission to redistribute their code). However, it makes it very difficult to make any changes to the license. On two occasions when it has been necessary to move to similar (but more modern or widely understood) licenses, we have simply made a good faith effort to contact all stakeholders that we are aware of, and proceeded on the basis that no objections were received. While unsatisfactory in some respects, this arrangement has actually worked well to date. Extensions to GAP were developed and expected to be used in much the same way as the rest of the system, and the existing license terms have been appropriate.

In the context of the wider OpenDreamKit ecosystem, including for instance datasets, on-line services and smart documents, this situation is less clear: we are working to build consensus in the GAP community around some new policies.

One example is the The Library of Small Groups. The main component of this is a dataset which lists and assigns unique IDs to all 423 million groups of order up to 2000 (except order 1024). The authors of this work were reluctant to release it under the GPL because of the risk to their professional reputations if someone released an "enhanced" version which contained mathematical errors, or altered the IDs so as to create confusion. After extensive discussion with the authors, an alternative free license — the Artistic License 2.0 was found which provided greater protections in their areas of concern, while being compatible with incorporation into OpenDreamKit tools.

Another example is the question of what constitutes a "derived work". This is a fundamental question since free software licensing is based wholly on copyright rather than patents as the basis for its IP. After discussion, the GAP community has taken the view that programs and packages written in the GAP language and smart documents based on the GAP Jupyter kernel or similar technology are not derived works: we make no attempt to control their redistribution or use, provided the included copy of GAP is properly sourced and acknowledged.

5.2. **Mutually incompatible software licenses**

Dated or mutually incompatible software licenses have long been a controversial subject in the open-source community. Yet another example is the custom license of the popular OpenSSL library for managing secure web connections. Its old license was famously incompatible with the GPL family of licenses, use by SAGE, creating some dubiousness as to whether it was legal

to distribute a copy of OpenSSL alongside SAGE, and thus doing so had been avoided, preferring to rely on a copy of OpenSSL already provided by the system. This approach was generally reliable on recent Linux distributions that maintained up-to-date copies of OpenSSL; Mac OS, on the other hand, also included a copy of OpenSSL for internal use but that was ill-maintained and not suitable for linking from arbitrary code.

This controversy impacted the SAGE project's and OpenDreamKit's aforementioned plans to promote development of new functionality outside the core package and distribute them on PyPI. Indeed, near the start of OpenDreamKit, PyPI switched to a policy of only allowing package downloads over secure SSL/TLS connections, thus requiring a library compatible with Python such as OpenSSL to be installed and up-to-date. This has created an unnecessary legal (but mostly not technical) challenge to promoting this new development model, as it made installation vastly more difficult for users, especially users on Mac OS, and forcing the developers to jump through hoops after hoops to try to circumvent the issue. Luckily, after a major relicensing effort, OpenSSL has finally switched to a more common and better-supported Apache license starting from its next release. This eventually will make it easier for software like SAGE to support it in the future.

A similar situation with the graphviz library has been a preventing developers from integrating it – and intermediate software layers such as dot2tex – in the SageMath distribution; this has been a continuous source of burden for end-users and their trainees.

With the advent of complex systems, the increasing barriers imposed by such incompatibilities has progressively encouraged software developers to use standard licensing schemes; in our communities, some consensus has organically emerged to use:

- GPL style licenses for mathematical software where ethical motivations dominate;
- BSD style licenses for general purpose tools such as Jupyter where practical motivations dominate, notably to facilitating their use and integration by industrial stakeholders.

♠**TO DO**: *@nchauvat: would you have some reflections to share about which licenses are more handy for companies?*♠

6. SUSTAINABILITY

In this final section, we review the different kinds of outcomes of OpenDreamKit, and assess their respective sustainability.

6.1. **Contributions to existing software**

There is a sustainability risk in writing software during a finite project period, as the end of the project can end funded support of the software. In all cases, the software that OpenDreamKit contributed to does not rely on infrastructure beyond free, public code hosting services such as GitHub. The sustainability is only vulnerable in that it requires human effort to continue further development and support.

Most of the work done by OpenDreamKit was contributing to existing packages, as opposed to starting completely new software packages. This software already had communities before OpenDreamKit started and we simply added some additional development effort. We expect that those communities will continue to maintain these packages after the project.

For example, during the four years of the OpenDreamKit project, 31398 commits were added to SAGE. Of these, 6194 (about one fifth) were made by OpenDreamKit members. This shows that there is a large community of contributors beyond OpenDreamKit.

In some cases, we reduced the sustainability burden by removing custom-built solutions and using more standard components. For example, our VRE is built upon Jupyter, which is widely used also outside of OpenDreamKit. This replaces the SAGE Notebook, which is no longer actively maintained.

6.2. **New software**

To a lesser extent, OpenDreamKit has also created several new software packages, such as nbdime, nbval, pypersist, various Jupyter kernels, ... In addition, cysignals and cypari2 existed as part of SAGE but were split off and made separate packages.

The typical sustainability model for open-source and community software is that software with sufficient interest and activity will develop a community of maintainers beyond the original authors, able to support the software as maintainers. In many cases, OpenDreamKit members will continue as maintainers of the software as community volunteers, or funded via other means. Some metrics for how well this is achieved is measuring contributions by individuals outside OpenDreamKit on a given project.

For WP4, the nbdime package was created as an official part of the Jupyter project, owned by the Jupyter organization on GitHub. This gives all members of the Jupyter team access to the repository to continue its maintenance. Jupyter is a large collaboration of many organizations, independent of OpenDreamKit. As of August, 2019, nbdime is installed from the Python Package Index (PyPI) on average 3,000 times per week. In the last twelve months, there have been 61 contributors to code and discussion, compared with 45 in the twelve months prior, indicating a growth in the community surrounding nbdime. In terms of code committed, contributions are still dominated by OpenDreamKit members, who will continue as maintainers of the project.

nbval is in a similar situation, owned by the Computational Modelling Group at University of Southampton / European XFEL, led by OpenDreamKit member Hans Fangohr. nbval is installed on average 3,500 times per week from PyPI. nbval has 25 contributors to code and discussion in the last twelve months, compared with 16 in the twelve months prior, and continues to be maintained by the Computational Modelling Group after the end of OpenDreamKit.

The packages cysignals and cypari2 were originally part of SAGE and will continue to be maintained by the SAGE community. This is a large community of developers, including multiple OpenDreamKit members.

6.3. **Prototypes and Symbolic Resources**

Work Package WP6 has fundamentally rethought the semantic aspects of VRE components and their interactions in mathematical workflows. The outcome of this was a refined concept of the aspects of "doing mathematics" which have to be covered by a VRE, as well as a semantic instantiation of the FAIR (Finable, Accessible, Interoperable, and Reusable) principles for mathematical data, knowledge, software, and (interactive) documents, that we call **deep FAIR**. In some cases, the new deep FAIR concepts could be retrofitted to OpenDreamKit systems (e.g. for schema/specification data in the LMFDB or by increasing the internal code organization in the GAP system), but in most cases, the concepts had to be implemented from scratch in dedicated research prototypes. These include

(1) the Math-in-the-Middle (MitM) Ontology,
(2) the System APIs (OMDoc/MMT theory graphs with alignments into the MitM Ontology) for GAP, Sage, LMFDB, Singular, and Pari/GP, with together about 10^5 constants, types, and constructors. These define system-specific OpenMath dialects.
(3) the OpenMath/SCSCP phrasebooks (input/output libraries for the OpenMath – in the respective system dialect defined by its System API theory graph) for GAP, Sage, LMFDB, Singular, and Pari/GP. Most of these are in fact packages that extend the systems.
(4) The MitM mediator that given the above can translate between the system dialects using the system APIs and alignments. This application builds on the MMT system [MMT].
(5) the `data.mathub.info` system, which extends MathHub for dataset management, this system takes MDDL dataset descriptions, a set of codecs, and provenance information to generate a database extension, a dataset importer, and a user interface. It is based on MMT [MMT] and Django [DJ] for database abstraction and management.

6.4. **Infrastructure setup**

OpenDreamKit components essentially benefited from setting up or further advancing the state of their regression testing, continuous integration, automation of artefact building and continuous deployment, examples of which can be found in deliverable D3.8: "Continuous integration platform for multi-platform build/test." or D5.15: "Final report and evaluation of all the GAP developments.". Such services are highly regarded by the communities around OpenDreamKit components due to their impact on their quality and sustainability.

We hope that the combination of such factors as using modern software engineering tools, public availability of their setup and build outcomes, and the value that they provide to respective communities will ensure that these services will continue to run and evolve, being maintained by volunteers or by other funded projects that may involve their maintenance.

6.5. **Dataset Production, Hosting, and Dissemination**

One of the main topics of WP6 has been studying the semantics of mathematical datasets as VRE components, using the OEIS, FindStat, and LMFDB as case studies. In the course of this endeavour OpenDreamKit

(1) has significantly improved the structure, maintainability, and interoperability of the LMFDB and
(2) developed a new MitM-interoperable model for deep FAIR mathematical datasets, which was implemented in the `data.mathub.info` system.

LMFDB is a large-scale management system for mathematical datasets in number theory, which is carried by a considerable and thriving research community independent of the OpenDreamKit project, which will ensure sustainability of the effort. OpenDreamKit has contributed the design and implementation of an online inventory for the LMFDB datasets, this inventory being itself stored as part of the LMFDB for ease of maintenance and for access by a variety of user interfaces. Building on this, ODK researchers have designed a new API for the LMFDB, with a Sage front-end still under development.

`data.mathub.info` is part of the MathHub.info system, which is a central research and development product of the KWARC group at FAU Erlangen-Nürnberg and development will be continued under this rubric. The OpenDreamKit project has motivated the system architecture, funded the development of the initial prototype, and triggered the inclusion of the first five data sets. The Math Data Workshop in August 2019 in Cernay has brought together a first community of enthusiasts which – so we hope – will carry the further development and use of the system beyond the FAU group. Indeed, the fledgling has its own slack channel and three new members and five additional data sets are already in the pipeline.

6.6. **OpenDreamKit's management infrastructure and web site**

Most of OpenDreamKit's management infrastructure is based on a dozen of mailing lists (hosted by french's public CNRS service), and a public GitHub organization `http://github.com/OpenDreamKit`. The content of the organization's repositories include documentation on the processes, the proposal, grant agreement, all reports, some papers, utilities to automatize many of the processes, communication material as well as the web site content including blog posts, workshop documents and other project activities. Tracking of OpenDreamKit's work packages, tasks and deliverables was carried over for the most part on the organization's issues. Up to a few dynamical goodies, OpenDreamKit's website at `opendreamkit.org` is build statically from its content, and hosted for free on the GitHub-pages service.

All this information will remain as an archive of the project activities and does not need much further updating. It is relevant for disseminating OpenDreamKit's achievements. The infrastructure setup is also open for reuse by other projects. More importantly, we hope it will inspire other projects to take on the same spirit! This archive may be of interest to people

conducting social studies on scientific communities developing software and infrastructure, and conducting open science in practice.

Mid term sustainability of the mailing lists should be guaranteed by CNRS. The GitHub organization is sustainable at no cost or effort as long as the GitHub service continues to exist, which is likely as it is a large platform with a long track record, currently owned by a large company. Should GitHub ever terminate its service, migration of all the content to other hosting services such as GitLab is possible with minimal efforts. In addition, for the very long run, all content hosted in repositories is archived indefinitely by the Software Heritage project (`https://www.softwareheritage.org/`). Being static, the website itself could easily be migrated to any other hosting service. There is no ongoing cost to host it beyond the negligible cost of renewing the `opendreamkit.org` domain (about 20 euros / year) which will be easily covered by the coordinators lab.

6.7. **Training and promotional material**

We took a modular approach to developing training materials, where the training materials for OpenDreamKit components were prepared (and taught) by the developers of the respective components. This approach ensured that they were prepared by experts, and that a training workshop can be easily assembled and taught by experts sourced from developer communities in a LEGO-style way from several courses. The composition of such courses will be made to match the level and needs of the audience, and the goals and time limits of by the workshop.

On the other hand, we made collegiate efforts to ensure that our training materials are coherent, and are updated as soon as possible to reflect changes in other OpenDreamKit components (as an example, documentation on LIBGAP was updated in GAP and SAGEMATH releases, separated by less than two months). Furthermore, teaching together gave us opportunity to observe each other's teaching and provide feedback both on instruction style and teaching materials. We strengthened links with The Carpentries, an global movement of volunteers who teach foundational coding and data science skills to researchers worldwide, and a number of project participants became officially certified Carpentries instructors before joining OpenDreamKit or over its duration, including Tania Allard, Erik Bray, Alexander Konovalov, Samuel Lelièvre and Michael Torpey (Tania Allard and Alexander Konovalov also became Trainers, qualified to teach new Carpentries instructors).

As an example of adopting Carpentries teaching practices, the Carpentries-style lesson "Programming with GAP" (`https://github.com/alex-konovalov/gap-lesson`), initiated by Alexander Konovalov in 2015 for the training events of the CCP (Collaborative Computational Project) CoDiMa subsequently was taught in 2018-2019, with the support of OpenDreamKit, at five training events by four instructors in various combinations, and has had three releases in 2016-2019 published on Zenodo `https://doi.org/10.5281/zenodo.597073`, with 9 new contributors.

At the occasion of the project, we also authored (with the help of experts) several explainer comics and life and motion-design videos. All the material is under creative commons licence, and we have seen the comics being reused in a variety of contexts to promote Binder.

Videos produced are currently hosted on YouTube which is safe in the short to mid run preservation, but a better option will have to be investigated for the long term, if still relevant then.

REFERENCES

[DJ] *Django – the Web Framework for Perfectionists with Deadlines*. URL: `https://www.djangoproject.com/` (visited on 08/29/2019).

[MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. project web site. URL: `https://uniformal.github.io/` (visited on 01/15/2019).

Disclaimer: this report, together with its annexes and the reports for the earlier deliverables, is self contained for auditing and reviewing purposes. Hyperlinks to external resources are meant as a convenience for casual readers wishing to follow our progress; such links have been checked for correctness at the time of submission of the deliverable, but there is no guarantee implied that they will remain valid.