Ref. Ares(2016)3367017 - 13/07/2016

EUROPEAN COMMISSION
Communications Networks, Contents & technology
eInfrastructure Science Cloud

ANNEX 1 (part A)

Research and Innovation action

NUMBER — 676541 — OpenDreamKit

Table of Contents

1.1. The project summary

Project Number [1]	676541	Project Acronym [2]	OpenDreamKit

One form per project

General information	
Project title [3]	Open Digital Research Environment Toolkit for the Advancement of Mathematics
Starting date [4]	01/09/2015
Duration in months [5]	48
Call (part) identifier [6]	H2020-EINFRA-2015-1
Topic	EINFRA-9-2015 e-Infrastructures for virtual research environments (VRE)
Fixed EC Keywords	Computer sciences, information science and bioinformatics, ENGINEERING AND TECHNOLOGY
Free keywords	pure mathematics,computer algebra,simulation,visualisation,component architecture,databases,reproducibility,Sage,iPython,Jupyter,LMFDB,MathHub

Abstract [7]

OpenDreamKit will deliver a flexible toolkit enabling research groupsto set up Virtual Research Environments, customised to meet the variedneeds of research projects in pure mathematics and applications andsupporting the full research life-cycle from exploration, throughproof and publication, to archival and sharing of data and code.OpenDreamKit will be built out of a sustainable ecosystem ofcommunity-developed open software, databases, and services, includingpopular tools such as LinBox, MPIR, Sage(sagemath.org), GAP, PariGP,LMFDB, and Singular. We will extend the Jupyter Notebook environmentto provide a flexible UI. By improving and unifying existing buildingblocks, OpenDreamKit will maximise both sustainability and impact,with beneficiaries extending to scientific computing, physics,chemistry, biology and more and including researchers, teachers, andindustrial practitioners.We will define a novel component-based VRE architecture and the adaptexisting mathematical software, databases, and UI components to workwell within it on varied platforms. Interfaces to standard HPC andgrid services will be built in. Our architecture will be informed byrecent research into the sociology of mathematical collaboration, soas to properly support actual research practice. The ease of set up,adaptability and global impact will be demonstrated in a variety ofdemonstrator VREs.We will ourselves study the social challenges associated withlarge-scale open source code development and of publications based onexecutable documents, to ensure sustainability.OpenDreamKit will be conducted by a Europe-wide demand-steeredcollaboration, including leading mathematicians, computationalresearchers, and software developers long track record of deliveringinnovative open source software solutions for their respectivecommunities. All produced code and tools will be open source.

1.2. List of Beneficiaries

Project Number [1]	676541	Project Acronym [2]	OpenDreamKit

List of Beneficiaries

No	Name	Short name	Country	Project entry date[8]	Project exit date
1	UNIVERSITE PARIS-SUD	UPSud	France		
2	CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE CNRS	CNRS	France		
3	JACOBS UNIVERSITY BREMEN GGMBH	JacobsUni	Germany		
4	UNIVERSITE GRENOBLE ALPES	UJF	France	01/01/2016	
5	TECHNISCHE UNIVERSITAET KAISERSLAUTERN	UNIKL	Germany		
6	THE CHANCELLOR, MASTERS AND SCHOLARS OF THE UNIVERSITY OF OXFORD	UOXF	United Kingdom		
7	UNIWERSYTET SLASKI	USlaski	Poland		
8	THE UNIVERSITY OF SHEFFIELD	USFD	United Kingdom		
9	UNIVERSITY OF SOUTHAMPTON	SOUTHAMPTON	United Kingdom		
10	THE UNIVERSITY COURT OF THE UNIVERSITY OF ST ANDREWS	USTAN	United Kingdom		
11	UNIVERSITE DE VERSAILLES SAINT-QUENTIN-EN-YVELINES.	UVSQ	France		
12	THE UNIVERSITY OF WARWICK	UWarwick	United Kingdom		
13	UNIVERSITAET ZUERICH	UZH	Switzerland		
14	LOGILAB	Logilab	France		
15	SIMULA RESEARCH LABORATORY AS	Simula	Norway		
16	UNIVERSITEIT GENT	UGent	Belgium	01/07/2016	

1.3. Workplan Tables - Detailed implementation

1.3.1. WT1 List of work packages

WP Number [9]	**WP Title**	**Lead beneficiary** [10]	**Person-months** [11]	**Start month** [12]	**End month** [13]
WP1	Project Management	1 - UPSud	55.00	1	48
WP2	Community Building, Training, Dissemination, Exploitation, and Outreach	1 - UPSud	122.00	1	48
WP3	Component Architecture	11 - UVSQ	118.00	1	48
WP4	User Interfaces	15 - Simula	154.00	1	36
WP5	High Performance Mathematical Computing	4 - UJF	200.00	1	48
WP6	Data/Knowledge/Software-Bases	3 - JacobsUni	132.00	1	48
WP7	Social Aspects	6 - UOXF	47.00	1	48
		Total	828.00		

1.3.2. WT2 list of deliverables

Deliverable Number [14]	**Deliverable Title**	**WP number** [9]	**Lead beneficiary**	**Type** [15]	**Dissemination level** [16]	**Due Date (in months)** [17]
D1.1	Basic project infrastructure (websites, wikis, issue trackers, mailing lists, repositories)	WP1	1 - UPSud	Websites, patents filling, etc.	Public	1
D1.2	Data Management Plan	WP1	1 - UPSud	ORDP: Open Research Data Pilot	Public	6
D1.3	Internal Progress Reports year 1	WP1	1 - UPSud	Report	Confidential, only for members of the consortium (including the Commission Services)	12
D1.4	Innovation Management Plan v1	WP1	1 - UPSud	Report	Confidential, only for members of the consortium (including the Commission Services)	18
D1.5	Internal Progress Reports v2	WP1	1 - UPSud	Report	Confidential, only for members of the consortium (including the Commission Services)	36
D1.6	Data Management Plan v2	WP1	1 - UPSud	ORDP: Open Research Data Pilot	Public	36
D1.7	Innovation Management Plan v2	WP1	1 - UPSud	Report	Confidential, only for members of the consortium (including the Commission Services)	45
D2.1	Starting press release	WP2	1 - UPSud	Websites, patents filling, etc.	Public	6
D2.2	Community building: Impact of development workshops, dissemination and	WP2	1 - UPSud	Report	Public	12

Deliverable Number [14]	Deliverable Title	WP number [9]	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
	training activities, year 1					
D2.3	Review on emerging technologies	WP2	1 - UPSud	Report	Public	12
D2.4	A short course for lecturers on using OpenDreamKit for delivering mathematical education.	WP2	8 - USFD	Websites, patents filling, etc.	Public	18
D2.5	Course material on using OpenDreamKit in data science	WP2	8 - USFD	Websites, patents filling, etc.	Public	24
D2.6	Community building: Impact of development workshops, dissemination and training activities, year 2	WP2	1 - UPSud	Report	Public	24
D2.7	Community-curated indexing service for OpenDreamKit	WP2	11 - UVSQ	Demonstrator	Public	24
D2.8	Demonstrator: Linear Algebra - interactive book	WP2	7 - USlaski	Demonstrator	Public	24
D2.9	Demonstrator: Nonlinear Processes in Biology interactive book	WP2	7 - USlaski	Demonstrator	Public	36
D2.10	Demonstrator: Interactive lecture notes and marking systems based on OpenDreamKit	WP2	8 - USFD	Demonstrator	Public	36
D2.11	Community building: Impact of development workshops, dissemination and training activities, year 3	WP2	1 - UPSud	Report	Public	36
D2.12	Demonstrator: Repository of interactive Notebooks in Machine Learning and Computational	WP2	8 - USFD	Demonstrator	Public	44

Deliverable Number [14]	**Deliverable Title**	**WP number** [9]	**Lead beneficiary**	**Type** [15]	**Dissemination level** [16]	**Due Date (in months)** [17]
	Biology based on OpenDreamKit					
D2.13	Micromagnetic VRE completed and online	WP2	9 - SOUTHAMPTON	Other	Public	48
D2.14	Demonstrators: Problems in Physics with Sage, Computational Mathematics for Engineering	WP2	7 - USlaski	Demonstrator	Public	47
D2.15	Community building: Impact of development workshops, dissemination and training activities, year 4	WP2	1 - UPSud	Report	Public	48
D2.16	Ending press release	WP2	1 - UPSud	Websites, patents filling, etc.	Public	48
D3.1	Virtual images and containers	WP3	11 - UVSQ	Other	Public	6
D3.2	Understand and document SAGEMATHCLOUD backend code.	WP3	1 - UPSud	Report	Public	18
D3.3	Support for the SCSCP interface protocol in all relevant components (SAGE, GAP, etc.) distribution	WP3	10 - USTAN	Other	Public	12
D3.4	Personal SAGEMATHCLOUD: single user version of SAGEMATHCLOUD distributed with SAGE.	WP3	1 - UPSud	Other	Public	24
D3.5	Integration between SAGEMATHCLOUD and SAGE's TRAC server	WP3	11 - UVSQ	Other	Public	24
D3.6	Open package repository for SAGE	WP3	11 - UVSQ	Other	Public	24
D3.7	One-click install SAGE distribution for Windows with Cygwin	WP3	1 - UPSud	Other	Public	24

Deliverable Number [14]	Deliverable Title	WP number [9]	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D3.8	Continuous integration platform for multi-platform build/test.	WP3	11 - UVSQ	Demonstrator	Public	36
D3.9	Semantic-aware SAGE interface to GAP.	WP3	6 - UOXF	Other	Public	36
D3.10	Packaging for major Linux distributions	WP3	11 - UVSQ	Other	Public	48
D3.11	HPC enabled SAGE distribution	WP3	4 - UJF	Other	Public	48
D4.1	Python/Cython bindings for PARI and its integration in Sage	WP4	2 - CNRS	Other	Public	6
D4.2	Active/Structured Documents Requirements and existing Solutions	WP4	3 - JacobsUni	Report	Public	9
D4.3	Distributed, Collaborative, Versioned Editing of Active Documents in MathHub.info	WP4	3 - JacobsUni	Demonstrator	Public	12
D4.4	Basic JUPYTER interface for GAP, PARI/GP, Singular	WP4	1 - UPSud	Other	Public	12
D4.5	SAGE notebook / JUPYTER notebook convergence	WP4	1 - UPSud	Demonstrator	Public	12
D4.6	Tools for collaborating on notebooks via version-control	WP4	15 - Simula	Other	Public	12
D4.7	Full JUPYTER interface for GAP, PARI/GP, SAGE, Singular	WP4	1 - UPSud	Other	Public	14
D4.8	Facilities for running notebooks as verification tests	WP4	15 - Simula	Other	Public	18
D4.9	In-place computation in active documents (context/computation)	WP4	3 - JacobsUni	Demonstrator	Public	18
D4.10	Second version of the PARI Python/Cython bindings	WP4	2 - CNRS	Other	Public	36

Deliverable Number [14]	Deliverable Title	WP number [9]	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D4.11	Notebook Import into MathHub.info (interactive display)	WP4	3 - JacobsUni	Demonstrator	Public	24
D4.12	JUPYTER extension for 3D visualisation	WP4	15 - Simula	Other	Public	24
D4.13	Refactorisation of SAGE's SPHINX documentation system	WP4	16 - UGent	Other	Public	24
D4.14	Computational Fluid dynamics visualisation in web notebook	WP4	15 - Simula	Other	Public	36
D4.15	Exploratory support for live notebook collaboration	WP4	15 - Simula	Other	Public	36
D4.16	Exploratory support for semantic-aware interactive widgets providing views on objects represented and or in databases	WP4	1 - UPSud	Demonstrator	Public	36
D5.1	Turn the Python prototypes for tree exploration into production code, integrate to SAGE.	WP5	1 - UPSud	Demonstrator	Public	3
D5.2	Facility to compile PYTHRAN compliant user kernels and Sage code and automatically take advantage of multi-cores and SIMD instruction units in CYTHON	WP5	4 - UJF	Demonstrator	Public	18
D5.3	Sun Grid Engine support for Project Jupyter Hub	WP5	8 - USFD	Other	Public	12
D5.4	Make PYTHRAN typing better to improve error information.	WP5	14 - Logilab	Demonstrator	Public	12
D5.5	Extend the existing assembly superoptimiser for AVX and upcoming Intel processor extensions for the MPIR library.	WP5	5 - UNIKL	Demonstrator	Public	18

Deliverable Number [14]	Deliverable Title	WP number [9]	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D5.6	Parallelise the relation sieving component of the Quadratic Sieve and implement a parallel version of Block-Wiederman linear algebra over GF2 and the triple large prime variant.	WP5	5 - UNIKL	Demonstrator	Public	18
D5.7	Take advantage of multiple cores in the matrix Fourier Algorithm component of the FFT for integer and polynomial arithmetic,and include assembly primitives for SIMD processor instructions (AVX, Knight’s Bridge, etc.), especially in the FFT butterflies	WP5	5 - UNIKL	Demonstrator	Public	18
D5.8	Explore the possibility to interface smoothly PYTHRAN, CYTHON and Cilk++	WP5	1 - UPSud	Demonstrator	Public	24
D5.9	Library design and domain specific language exposing LINBOX parallel features to SAGE	WP5	4 - UJF	Report	Public	24
D5.10	Devise a generic parallelisation engine for PARI and use it to prototype selected functions (integer factorisation, discrete logarithm, modular polynomials)	WP5	2 - CNRS	Demonstrator	Public	24
D5.11	Refactor and Optimise the existing combinatorics SAGE code using the new developed PYTHRAN and CYTHON features	WP5	2 - CNRS	Demonstrator	Public	36
D5.12	Exact linear algebra algorithms and implementations. Library maintenance and close integration	WP5	4 - UJF	Demonstrator	Public	36

Deliverable Number [14]	Deliverable Title	WP number [9]	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
	in mathematical software for LINBOX library					
D5.13	Parallelise the Singular sparse polynomial multiplication algorithms and provide parallel versions of the Singular sparse polynomial division and GCD algorithms.	WP5	5 - UNIKL	Demonstrator	Public	48
D5.14	Implementations of exact linear algebra algorithms on distributed memory et heterogenous architectures: clusters and accelerators. Solving large linear systems over the rationals is the target application.	WP5	4 - UJF	Demonstrator	Public	48
D5.15	Final report and evaluation of the GAP developments	WP5	10 - USTAN	Report	Public	48
D5.16	PARI suite release (LIBPARI, GP and GP2C) that fully support parallelisation allowing individual implementations to scale gracefully between single core / multicore / massively parallel machines.	WP5	2 - CNRS	Demonstrator	Public	48
D6.1	Full-text Search (Formulae + Keywords) over LaTeX-based Documents (e.g. the arXiv subset)	WP6	3 - JacobsUni	Other	Public	9
D6.2	Initial DKS base Design (including base survey and Requirements Workshop Report)	WP6	3 - JacobsUni	Report	Public	12
D6.3	Design and implementation	WP6	3 - JacobsUni	Other	Public	15

Deliverable Number [14]	**Deliverable Title**	**WP number** [9]	**Lead beneficiary**	**Type** [15]	**Dissemination level** [16]	**Due Date (in months)** [17]
	of Triform (DKS) Theories					
D6.4	Conversion of existing and new Databases (LMFDB, OEIS,FindStat) to unified interoperable System	WP6	13 - UZH	Websites, patents filling, etc.	Public	24
D6.5	PYTHON/SAGE Computational Foundation Module in OMDoc/MMT	WP6	3 - JacobsUni	Other	Public	24
D6.6	Full-text search (Formulae + Keywords) over CAS Modules and Notebooks	WP6	3 - JacobsUni	Other	Public	30
D6.7	PYTHON/SAGE Declarative Semantics in OMDoc/MMT	WP6	3 - JacobsUni	Other	Public	36
D6.8	LMFDB Algorithm Verification with respect to a Triformal Theory	WP6	3 - JacobsUni	Other	Public	36
D6.9	Shared persistent Memoisation Library for PYTHON/SAGE	WP6	10 - USTAN	Other	Public	42
D6.10	Search from Notebooks/Active Documents Interface	WP6	3 - JacobsUni	Other	Public	42
D6.11	LMFDB Integration of Algorithms, Data and Presentation	WP6	3 - JacobsUni	Report	Public	48
D7.1	The flow of code and patches in open source projects	WP7	6 - UOXF	Report	Public	18
D7.2	TRAC add-on to manage ticket prioritisation	WP7	6 - UOXF	Other	Public	24
D7.3	Demo: Mechanism for comments on posted Jupyter notebooks	WP7	8 - USFD	Demonstrator	Public	24
D7.4	Demo: Jupyter Notebook Live Poster	WP7	8 - USFD	Demonstrator	Public	36
D7.5	Report on relevant research in sociology of mathematics and	WP7	6 - UOXF	Report	Public	42

Deliverable Number [14]	**Deliverable Title**	**WP number** [9]	**Lead beneficiary**	**Type** [15]	**Dissemination level** [16]	**Due Date (in months)** [17]
	lessons for design of OpenDreamKit VRE					
D7.6	Review of new publication mechanisms, including evaluation of demonstrator projects	WP7	8 - USFD	Report	Public	42
D7.7	Game-theoretic analysis of development practices in open-source VREs	WP7	6 - UOXF	Report	Public	42
D7.8	Micromagnetic VRE environment evaluation report	WP7	9 - SOUTHAMPTON	Report	Public	48

1.3.3. WT3 Work package descriptions

Work package number [9]	WP1	**Lead beneficiary** [10]	1 - UPSud
Work package title	Project Management		
Start month	1	**End month**	48

Objectives

Establish and maintain an effective contract, project, and operational management approach, ensuring
(i) an effective and timely implementation of the project,
(ii) quality control of the results,
(iii) risk and innovation management of the project as a whole as well as
(iv) timely and necessary interaction with the EC and other interested parties.

Description of work and role of partners

WP1 - Project Management [Months: 1-48]
UPSud, CNRS, JacobsUni, UJF, UNIKL, UOXF, USlaski, USFD, SOUTHAMPTON, USTAN, UVSQ, UWarwick, UZH, Logilab, Simula, UGent
The project will be managed by UPS, which has extensive experience in administering and leading EU funded and na- tional projects. The coordinator together with the WP leaders, will be responsible for monitoring WP status, coordination of work plan updates and annual internal progress reports. The project management structure and roles of partners in the consortium are presented in Section 3.2.

T1.1 Project and financial management M0-M48@.2; M0-M3; M10-M12; M22-M24; M34-M36; M42-M48
Sites: UPSud (lead), Logilab, UVSQ, UJF, CNRS, UOXF, USFD, SOUTHAMPTON, USTAN, UWarwick, JacobsUni, UNIKL, USlaski, UZH, Simula, UGent

The task includes the following activities
• Preparation and Distribution of the Consortium Agreement;
• Setting up the project website, intranet and communication procedures for effective communication;
• Organisation of project review and progress meetings;
• Establishment and maintenance of external contacts (with the EC, other relevant national / EU projects, other academic and industrial stakeholders) to organise transfer of knowledge, present and promote project results;
• Progress and Financial Reporting to the EC;
• Data and IPR Management will be managed in accordance with agreed rules stated in the Consortium Agreement and in accordance with the Data Management Plans (D1.2,D1.6).

T1.2 Quality assurance and risk management M6-M48@.3
Sites: UPSud (lead), Logilab, UVSQ, UJF, CNRS, UOXF, USFD, SOUTHAMPTON, USTAN, UWarwick, JacobsUni, UNIKL, USlaski, UZH, Simula, UGent

A quality assurance plan will be established to ensure coherent and sufficient quality of the work and results. The plan will be developed by UPS, involving all partners, and will be followed up regularly. In addition, the project coordinator with support from the coordination team and quality review board will establish and review annually a risk management plan and self-assessment to ensure that technical barriers / potential risks are identified and corrective measures are put into place on time (D1.3).

T1.3 Innovation management M6-M48@.2
Sites: UPSud (lead), Logilab, UVSQ, UJF, CNRS, UOXF, USFD, SOUTHAMPTON, USTAN, UWarwick, JacobsUni, UNIKL, USlaski, UZH, Simula, UGent

One of the most important criteria for success for the OpenDreamKit project is to bring the project results into use. Therefore, exploitation routes will be sought whenever possible. In order to create a common understanding within the Consortium of how we can best shepherd an idea all the way from conception to realisation and exploitation, the Coordinator will be responsible for the preparation and realisation of an Innovation Plan. This plan will assure that research activities meet the required milestones and produce outputs fully aligned with the project objectives. All research activities will go through an initial process where the exploitation opportunity is identified along with the main stakeholders for the exploitation opportunity and an IP owner.

Participation per Partner

Partner number and short name	**WP1 effort**
1 - UPSud	26.50
2 - CNRS	2.00
3 - JacobsUni	2.00
4 - UJF	2.00
5 - UNIKL	2.00
6 - UOXF	2.00
7 - USlaski	2.00
8 - USFD	2.00
9 - SOUTHAMPTON	2.00
10 - USTAN	2.00
11 - UVSQ	2.00
12 - UWarwick	2.00
13 - UZH	1.00
14 - Logilab	2.00
15 - Simula	2.00
16 - UGent	1.50
Total	55.00

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D1.1	Basic project infrastructure (websites, wikis, issue trackers, mailing lists, repositories)	1 - UPSud	Websites, patents filling, etc.	Public	1
D1.2	Data Management Plan	1 - UPSud	ORDP: Open Research Data Pilot	Public	6
D1.3	Internal Progress Reports year 1	1 - UPSud	Report	Confidential, only for members of the consortium (including the Commission Services)	12
D1.4	Innovation Management Plan v1	1 - UPSud	Report	Confidential, only for members of the consortium (including the Commission Services)	18
D1.5	Internal Progress Reports v2	1 - UPSud	Report	Confidential, only for members of the consortium (including the Commission Services)	36
D1.6	Data Management Plan v2	1 - UPSud	ORDP: Open Research Data Pilot	Public	36
D1.7	Innovation Management Plan v2	1 - UPSud	Report	Confidential, only for members of the consortium (including the Commission Services)	45

Description of deliverables

D1.1 : Basic project infrastructure (websites, wikis, issue trackers, mailing lists, repositories) [1]
Basic project infrastructure (websites, wikis, issuetrackers, mailing lists, repositories)

D1.2 : Data Management Plan [6]
Data Management Plan

D1.3 : Internal Progress Reports year 1 [12]
Internal Progress Reports, including risk management and quality assurance plan, year1

D1.4 : Innovation Management Plan v1 [18]
Innovation Management Plan v1

D1.5 : Internal Progress Reports v2 [36]
Internal Progress Reports, including risk management and quality assurance plan v2

D1.6 : Data Management Plan v2 [36]
Data Management Plan v2

D1.7 : Innovation Management Plan v2 [45]
Innovation Management Plan v2

Schedule of relevant Milestones

Milestone number [18]	**Milestone title**	**Lead beneficiary**	**Due Date (in months)**	**Means of verification**
MS1	Startup	1 - UPSud	12	By Milestone 1 we will have carried out the requirements study, design and prototype implementations and started community building activities.
MS2	Implementations	1 - UPSud	24	By Milestone 2 we will have constructed first fully functional interface implementations and released enhanced versions of OpenDreamKit components, and train early adopters of Open-DreamKit.
MS3	Community/ Experiments	1 - UPSud	36	By Milestone 3 we will have gathered and evaluated feedback on OpenDreamKit software and established the portfolio of experiments produced with OpenDreamKit through further engaging with the community.
MS4	Evaluation	1 - UPSud	48	By Milestone 4 we will have released final versions of all OpenDreamKit components and completed the project evaluation.

Work package number [9]	WP2	**Lead beneficiary** [10]	1 - UPSud
Work package title	Community Building, Training, Dissemination, Exploitation, and Outreach		
Start month	1	**End month**	48

Objectives

The objective of this work package is to further develop the community at the European scale, foster cross team collaboration, spread the expertise, and engage the greater community to participate in the definition and refinement of the
requirements, and the implementation and use of the produced solutions. This includes:
• ensuring awareness of the results in the user community;
• engaging cross communities discussions to foster scientific collaboration and conjoint development;
• spreading the expertise through workshops and training sessions;
• providing training for partners inside the project, the community engaging with contributions to the project, and end-users of OpenDreamKit
• reviewing emerging technologies,
• develop demonstrators,
• defining individual exploitation plans; and,
• managing existing and new intellectual property.

Description of work and role of partners

WP2 - Community Building, Training, Dissemination, Exploitation, and Outreach [Months: 1-48]
UPSud, CNRS, UNIKL, USlaski, USFD, SOUTHAMPTON, USTAN, UVSQ, Logilab, Simula, UGent
We will organise regular open workshops (e.g. Sage Days, PARI Days, summer schools, etc.); some of them will be focused on development and coding sprints, and others on training. This is also an occasion to organise cross community workshops like Sage-Jupyter days.

Some of the networking activities and internal training will come from short- to long-term mutual visits by participants, to collaborate on specific features. A typical such visit would bring together an JUPYTER developer with a GAP developer for a couple of days to implement a first prototype of a notebook interface to GAP. A number of demonstrators will be developed in the project and disseminated in different ways. We will also participate in the concertation activities, consultations and other meetings and events of the European E-Infrastructure projects.

All the code, documents, test and build infrastructure produced as part of the project will be made available as open source. Open access to all publications resulting from the project will be ensured. This work package will complement and lean on a parallel COST network which role is to build and engage a greater community.

T2.1 Dissemination and Communication activities M0-M48
Sites: UPSud (lead), USTAN

This task comprises all forms of direct dissemination and public communication activities such as press releases, creation of the project web-site (D1.7) including visitor analysis and monitoring tools, scientific and technical publications, outreach activities (seminars, keynote talks, media interviews, press releases), pro-motion through social media (e.g. Twitter, Facebook, LinkedIn), creation of advertisement materials such as flyers, posters, and electronic feeds as well as their distribution. We will use standard community building technology such as mailing lists, Wikis and Forums, to ensure dissemination and engagement of the community to support

this. We will also generate press releases at appropriate moments
(D2.1, D2.16).

T2.2 Training and training portal M0-M48@.1; M0-M1
Sites: UPSud (lead)

Training is at the heart of this project: through our open source approach, networking activities, workshops, demonstra- tors (T2.10), interactive books (such as in T4.13 and T2.9), and training for teachers and trainers (T2.6), we have firmly integrated the training aspect into the core of our project plan. Each of these activities will create dedicated webpages hosting the material to make it accessible to public as wide as possible — in line with our philosophy for software, we believe in the benefits of sharing the material and maximising the value of the financial investment into this project.

In this task, we create a central OpenDreamKit training portal that serves as an inclusive point of entry to explore available training materials. This will be hosted on the projects website (T2.1).

T2.3 Community Building: Development Workshops M0-M48
Sites: UPSud (lead), CNRS, UNIKL, Simula, USTAN, USFD, UGent

We will organise development workshops all throughout the project. The aim of these workshops is to bring together developers from the different communities to design and implement some key aspects of OpenDreamKit such as user interface, and documentation and to ensure cross compatibility. These meetings will gather not only participants of OpenDreamKit but also members of the different communities involved. Bringing talented people together is the best way to make actual progress on the different aspects of OpenDreamKit and to kick-start new challenges. It is also a way to work within the communities we're reaching in and include them in the discussions and development. It fosters collaboration between scientists and developers from different backgrounds to build tools that are needed by all.

Each workshop will be aimed at a specific software component (SAGE , GAP, SAGE MATH CLOUD , IPYTHON , SINGULAR , etc.) or be joint meeting between different communities to improve interoperability and joint developments. We are planning to have 4 or 5 such events per year. The currently anticipated list of workshops includes:

• One SAGE workshop per year in Cernay France (near Orsay) where similar gatherings took place before. One of them will be a Sage-Sphinx day, dedicated to documentation.

• One Atelier PARI in Bordeaux per year. The team in Bordeaux has a great experience in organizing this kind of PARI events.

• Two SINGULAR workshops and two GAP-S INGULAR workshops in Kaiserslautern over the four years.

• Two workshops dedicated to high performance mathematical computing in relations with WP5. One of them should be in Grenoble and the second one in Bordeaux to foster the work with PARI towards T5.1.

• Two Data Science workshops which use OpenDreamKit to develop effective machine learning and data modelling practice organised by Sheffield.

• A joint meeting on the topics of SAGE MATH CLOUD and JUPYTER in Simula in relation with WP4.
• A joint event between GAP, SAGE , and SINGULAR in ICMS, Edinburgh.
• A joint JUPYTER and SAGE event in Orsay.

• A joint LMFDB and SAGE event in Warwick to work towards WP6.

Yearly reports on the impact of these workshops on the community (D2.2, D2.6, D2.11, D2.15) will be delivered.

T2.4 Reviewing emerging technologies M0-M48
Sites: UPSud (lead), USTAN, SOUTHAMPTON, USFD, USlaski,UVSQ, CNRS, Simula

In this task, we will produce periodic reviews (D2.3) of emerging technologies and relevant developments elsewhere, and implications for our plans, taking into account input from the communities. This include the review of standard components and service for storage and sharing, computational resources, authentication, package management, etc. This may further include negotiating access or shared development when appropriate. This information will be fed to the other work packages, in particular Work Package WP3 Component Architecture.

T2.5 Dissemination: reaching towards users and fostering diversity M0-M48
Sites: UPSud (lead), CNRS, USFD, USTAN

As lead developers of OpenDreamKit, most of us consider themselves as both scientists and developers. We have experience in reducing the gap between those two worlds. We organise training, workshops such as Sage-days to promote our tools and bring more users and developers from the scientific world. On the other hand, we often at- tend and present to more development-oriented gatherings like PyCon and SciPy to interact with engineers and foster collaboration.

The aim of this task is to exploit this winning strategy within OpenDreamKit. Three events should be organised in the spirit of Sage-days to gather and train more users and foster scientific development around OpenDreamKit. These conferences usually welcome around 50 participants and have a big impact on the scientific community. One of them will be at CIRM in Marseille, another one at ICMS in Edinburgh and a third one probably in Dagstuhl, Germany. In the same spirit, we will also have training sessions organised within the universities (Orsay and Grenoble). We will also run a series of 4 workshops in developing countries, especially in Africa and South America. Some of these workshops will be connected to CIMPA schools. The CIMPA is an international organisation based in Nice (France) that promote research in mathematics in developing countries. It organises each year around 20 schools. The under-representation of women in the scientific world is even more visible if we intersect science with software development. As we know, we have many talented women in our community, and we will organise some events targeted at women in the spirit of the ”Women in Sage” days that happened many times in the US already. We are planning to have two of them in Orsay and at least one in Oxford where ”Women in CS” days already take place on a regular basis. Apart from these different events, we will also be present at important events of both our scientific community (international mathematical conferences such as FPSAC for combinatorics) and the python / open-source software development community: PyCon, SciPy, EuroPython, etc. The material we develop for presentation at these events will be made publicly available.

T2.6 Introduce OpenDreamKit to Researchers and Teachers M6-M44
Sites: USFD (lead), SOUTHAMPTON

In this task, we will develop and deliver materials that will introduce OpenDreamKit to potential users—both researchers and teachers. Our initial focus will be on teachers, but as the results from WP7 become available we will deploy them with researchers, both local to the University of Sheffield and across the wider computational biology and machine learning fields.

We will develop a ‘taster’ seminar (1-2 hours) and follow-up short course (1-2 days) on OpenDreamKit for researchers and lecturers in all disciplines D2.4. At Sheffield, this will be added to the set of courses that are offered as part of IT Services’ research support department. As such, it could potentially reach all disciplines. It will also be made publicly available for widespread dissemination and collaborative modification. Elements of this work will also be integrated with the GP Summer Schools and Roadshows (http://ml.dcs.shef. ac.uk/gpss/). The Summer School is now in its fourth edition (over 140 students educated). The Roadshows have taken place in Uganda, Colombia and in 2015 they are scheduled for Italy, Australia and Kenya. The Kenya school will be the first to have more of a ‘data science’ focus that we think will be particularly appropriate for dissemination of OpenDreamKit (D2.5).

These seminars and short courses will also be used to identify potential collaborators who are interested in utilising OpenDreamKit immediately. We will act as consultants to these collaborators in two ways: We will work with lecturers at Sheffield to introduce OpenDreamKit to various disciplines via the production of interactive lecture notes (D2.10). The focus for the student here will not necessarily be on programming but rather on interaction with the subject matter via use of OpenDreamKit. 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. We will provide technical and programming expertise to lecturers—helping them to develop the interactive part of notes while they provide the subject material.

We will work as consultants with researchers at Sheffield to introduce OpenDreamKit to their workflow. Any projects that successfully do this will be promoted as case studies for OpenDreamKit. Finally, our aim will be to introduce ideas from WP7 in our teaching materials. By the end of the project we will have produced a series of interactive notebook demonstrators D2.12 of OpenDreamKit with a particular focus on computational biology, data science and machine learning. These notebooks will expand the use of VREs in these domains, appealing to researchers used to the domains of Bioconductor and MATLAB. We will make use of live notebook posters (D7.4) and commenting systems (D7.3). These interactive notebooks will be provided in a public repository (D2.12).

T2.7 Open source dissemination of micromagnetic VRE M28-M32
Sites: SOUTHAMPTON (lead), Simula, USFD,UPSud

Tasks T4.11 and T4.13 provide the micromagnetic VRE demonstrator (1.3.7) built on top of OpenDreamKit. In this task, we set up of the infrastructure to encourage and invite code contributions from the micromagnetic community to both code and created notebooks, while automating quality control and maintaining trust effectively (D2.13). The source code of the micromagnetic VRE will be made available as open source on public repository hosting sites (such as GitHub/Bitbucket), and announced to the community via appropriate mailing lists and other means. We will set up a publicly accessible Jenkins/Travis continuous integration (CI) system to (i) run regression tests (from T3.8 and T4.11) routinely when the micromagnetic VRE code or underlying OOMMF core code changes, (ii) re-execute note- books (from T4.13) and use them as regression tests (using the outcome of task T4.3), and (iii) re-build downloadable installation files and virtual machine images.

T2.8 Micromagnetic VRE dissemination workshops M18-M44@0.23
Sites: SOUTHAMPTON (lead)
We will run a series of workshops (D2.13) during the evenings of 4 major international meetings on magnetism re- search 8 to disseminate the micromagnetic virtual research environment (Sect. 1.3.7 and T2.7) in the micromagnetic community. Each conference attracts around 1500 participants, and we expect at least 30 for our workshops at every event. Depending on demand, multiple workshops will be given per conference. The taught material will include (i) use of the J UPYTER -based micromagnetic VRE, and an (ii) introduction to the stan- dard techniques for contributing to open source software (version control, pull requests, testing frameworks) to foster excellence in computational science and to make the micromagnetic VRE project self-sustaining as quickly as possible. In addition, all teaching materials, including videos, will be made available on a website. For each workshop, we request e 500 room hire at the magnetism conference location and the travel expenses for two teachers from Southampton to attend the one week international conference, totalling (e 500 + 2x e 2200= e 4900) per workshop. There are no other costs.

T2.9 Demonstrator: Interactive books M0-M46
Sites: USlaski (lead), SOUTHAMPTON
One of the important elements of VREs is a common flexible writing format which enables the creation of large structured documents. There are many known solutions to that problem, but they usually compromise the interactivity of the notebook interface and typesetting quality of desktop publishing software like LaTeX. Recently, a few approaches tried to bring both interactivity and the typographic features. The modestly tagged markup language DocOnce targets the problem of reusability of the document source code for producing traditional LaTeX-based printed books, IPython notebooks, Sphinx documents (with Sage cells), and many other formats. MathBook XML is a lightweight XML application for authors of scientific articles, textbooks and monographs extensively using Sage cells for interactive elements. The Sphinx documentation software has been successfully applied for creation of interactive books containing Sage cells. Additional interactivity is offered using the Runestone tools. The technical aspects of format for interactive publications is a subject of the task “Structured documents” in T4.6. In this task we will demonstrate usability of the results of T4.6 in creation of scientific textbooks. Three interactive books will be created:

- Nonlinear Processes in Biology (D2.9)
- Linear Algebra (D2.8)
- Computational Mathematics for Engineering (D2.14)
- Problems in Physics with Sage/Python (D2.14)

The choice of those particular topics has been made for the sake of maximal diversity. The “Nonlinear Processes in Biology” will heavily use numerical solution of ODEs and PDEs, the Linear Algebra book will be a classical mathematical textbook, while the next book is targetting the engineering community. The last example will focus mostly on collab- orative editing and modularity of content which is produced using VRE technologies. We will demonstrate the power of symbolic algebra where appropriate throughout. All books are natively designed within the VRE, pushing the next generation of learning methodology out into multiple communities.

The main research aspect for this task will be to integrate modern computational tools in classical scientific topics and explore how the VRE environment can accelerate the development and produce electronic documents with significantly enhanced pedagogy and learning effectiveness.

In particular we will answer following questions:
• When is a fully interactive worksheet required and when is a textbook with executable code cells sufficient?
• How to assemble a classical monograph by reusing independently working building block of text and code?
• What are best tools and practices for using a single source for producing printed and electronic (interactive) textbooks?
• How to collaboratively write reusable course material?
• How can we facilitate automatic testing of all code examples, plots, etc?
• How can students can benefit from using VRE?

T2.10 Demonstrator: Computational mathematics resources indexing service M20-M25 Sites: UVSQ (lead), CNRS Beyond official documentation and tutorials, users of mathematical software and VREs learn from a wide array of sources: university courses, Q& A sites, web searches, etc. A simple web search on any major software component yields dozens of non-official tutorials and how-tos in many different languages. However, search engines mostly miss the relevant metadata: how does one find a tutorial on linear algebra in PARI/GP, written at an undegraduate level, in French or Spanish?

This need has been felt by most communities at some point, and each has come up with its own solution: most soft- ware components (e.g., GAP, PARI/GP, SAGE , . . .) simply link material from their official page; SAGE has a wiki (http://wiki.sagemath.org/) referencing additional resources, and used to host a large number of tutorial work- sheets on http://sagenb.org/ ; the recent introduction of public projects in SAGE MATH CLOUD is sparking approxi- mately the same phenomenon that had previously happened with http://sagenb.org/ ; IPYTHON host the Notebook Viewer service (http://nbviewer.ipython.org/), which renders (without hosting) community-made notebooks; and teaching institutions host or link their own collections of pedagogical resources. These collections are usually incomplete, limited in scope, hard to search, and difficult to keep up-to-date. What the community needs is a community-curated, searchable, metadata-driven, multilingual, platform agnostic indexing service whose goal is to reference and rank all the community generated knowledge around a software component or VRE. The goal of this task is to create the tool powering such service, and to host a (free) community-curated index for OpenDreamKit related resources as a demonstrator (D2.7).

Participation per Partner

Partner number and short name	WP2 effort
1 - UPSud	9.00
2 - CNRS	19.00
5 - UNIKL	2.00

Partner number and short name	WP2 effort
7 - USlaski	30.00
8 - USFD	17.00
9 - SOUTHAMPTON	16.00
10 - USTAN	18.00
11 - UVSQ	2.00
14 - Logilab	6.00
15 - Simula	2.00
16 - UGent	1.00
Total	122.00

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D2.1	Starting press release	1 - UPSud	Websites, patents filling, etc.	Public	6
D2.2	Community building: Impact of development workshops, dissemination and training activities, year 1	1 - UPSud	Report	Public	12
D2.3	Review on emerging technologies	1 - UPSud	Report	Public	12
D2.4	A short course for lecturers on using OpenDreamKit for delivering mathematical education.	8 - USFD	Websites, patents filling, etc.	Public	18
D2.5	Course material on using OpenDreamKit in data science	8 - USFD	Websites, patents filling, etc.	Public	24
D2.6	Community building: Impact of development workshops, dissemination and training activities, year 2	1 - UPSud	Report	Public	24
D2.7	Community-curated indexing	11 - UVSQ	Demonstrator	Public	24

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
	service for OpenDreamKit				
D2.8	Demonstrator: Linear Algebra - interactive book	7 - USlaski	Demonstrator	Public	24
D2.9	Demonstrator: Nonlinear Processes in Biology interactive book	7 - USlaski	Demonstrator	Public	36
D2.10	Demonstrator: Interactive lecture notes and marking systems based on OpenDreamKit	8 - USFD	Demonstrator	Public	36
D2.11	Community building: Impact of development workshops, dissemination and training activities, year 3	1 - UPSud	Report	Public	36
D2.12	Demonstrator: Repository of interactive Notebooks in Machine Learning and Computational Biology based on OpenDreamKit	8 - USFD	Demonstrator	Public	44
D2.13	Micromagnetic VRE completed and online	9 - SOUTHAMPTON	Other	Public	48
D2.14	Demonstrators: Problems in Physics with Sage, Computational Mathematics for Engineering	7 - USlaski	Demonstrator	Public	47
D2.15	Community building: Impact of development workshops, dissemination and training activities, year 4	1 - UPSud	Report	Public	48

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D2.16	Ending press release	1 - UPSud	Websites, patents filling, etc.	Public	48

Description of deliverables

D2.1 : Starting press release [6]
Starting press release

D2.2 : Community building: Impact of development workshops, dissemination and training activities, year 1 [12]
Community building: Impact of development workshops, dissemination and training activities, year 1

D2.3 : Review on emerging technologies [12]
Review on emerging technologies

D2.4 : A short course for lecturers on using OpenDreamKit for delivering mathematical education. [18]
A short course for lecturers on using OpenDreamKit for delivering mathematical education.

D2.5 : Course material on using OpenDreamKit in data science [24]
Course material on using OpenDreamKit in data science.

D2.6 : Community building: Impact of development workshops, dissemination and training activities, year 2 [24]
Community building: Impact of development workshops, dissemination and training activities, year 2

D2.7 : Community-curated indexing service for OpenDreamKit [24]
Community-curated indexing tool and service for OpenDreamKit

D2.8 : Demonstrator: Linear Algebra - interactive book [24]
Demonstrator: Linear Algebra - interactive book

D2.9 : Demonstrator: Nonlinear Processes in Biology interactive book [36]
Demonstrator: Nonlinear Processes in Biology interactive book

D2.10 : Demonstrator: Interactive lecture notes and marking systems based on OpenDreamKit [36]
Demonstrator: Interactive lecture notes and marking systems based on OpenDreamKit.

D2.11 : Community building: Impact of development workshops, dissemination and training activities, year 3 [36]
Community building: Impact of development workshops, dissemination and training activities, year 3

D2.12 : Demonstrator: Repository of interactive Notebooks in Machine Learning and Computational Biology based on OpenDreamKit [44]
Demonstrator: Repository of interactive Notebooks in Machine Learning and Computational Biology based on OpenDreamKit

D2.13 : Micromagnetic VRE completed and online [48]
Micromagnetic VRE completed and online

D2.14 : Demonstrators: Problems in Physics with Sage, Computational Mathematics for Engineering [47]
Demonstrators: Problems in Physics with Sage, Computational Mathematics for Engineering

D2.15 : Community building: Impact of development workshops, dissemination and training activities, year 4 [48]
Community building: Impact of development workshops, dissemination and training activities, year 4

D2.16 : Ending press release [48]
Ending press release

Schedule of relevant Milestones

Milestone number [18]	Milestone title	Lead beneficiary	Due Date (in months)	Means of verification
MS1	Startup	1 - UPSud	12	By Milestone 1 we will have carried out the requirements study, design and prototype implementations and started community building activities.
MS2	Implementations	1 - UPSud	24	By Milestone 2 we will have constructed first fully functional interface implementations and released enhanced versions of OpenDreamKit components, and train early adopters of Open-DreamKit.
MS3	Community/ Experiments	1 - UPSud	36	By Milestone 3 we will have gathered and evaluated feedback on OpenDreamKit software and established the portfolio of experiments produced with OpenDreamKit through further engaging with the community.
MS4	Evaluation	1 - UPSud	48	By Milestone 4 we will have released final versions of all OpenDreamKit components and completed the project evaluation.

Work package number [9]	WP3	**Lead beneficiary** [10]	11 - UVSQ
Work package title	Component Architecture		
Start month	1	**End month**	48

Objectives

The objective of this work package is to develop and demonstrate a set of APIs that enable components, such as database interfaces, computational modules, separate systems such as GAP or SAGE, to be flexibly combined and run smoothly across a wide range of environments (such as Cloud-based, local, and server environments).

Description of work and role of partners

WP3 - Component Architecture [Months: 1-48]
UVSQ, UPSud, UJF, UOXF, SOUTHAMPTON, USTAN, Logilab, UGent
This Work Package focuses on the structure of the components that make up a mathematical software and their interactions. Such components can be separate modules inside a unique software, or separate softwares interacting through library calls and/or through APIs (e.g.: web APIs). When combined together, they make up a full VRE. The architecture of these software components must be:

• Portable, to support a wide range of platforms (mobile, desktops, cloud, ...).
• Modular, so to ease installing, building, testing, and remixing.

• Flexible, so to adapt to different use cases: personal computation, HPC, parallel platforms, . . .

• Open, in the sense of open source, but also in the sense of clearly documented and open to the user who wants to understand its underpinnings and/or contribute to it. Indeed we must not forget that the working mathematicians and other users need to know what algorithms the software is going to run to solve a given problem.

T3.1 Portability M0-M36
Sites: UVSQ (lead), UPSud, UGent

In order to achieve maximum availability and accessibility, mathematical software must be developed and tested for a wide range of computer architectures and operating systems. However most of open source development happens in POSIX environments (usually Linux or OS X), and almost exclusively on x86 platforms. The vast majority of the developers of mathematical software does not have the expertise, nor the access to appropriate hardware and software, to insure appropriate testing and porting of components. The best incarnation of this issue is the involved installation procedure for SAGE on Windows, a major adoption barrier and common source of complaints by end-users. In this task we will address the common needs of the community in terms of portability layers, building and testing infrastructure.

• Best practices adopted by the larger open source community will be investigated and leveraged, and existing expertise will be shared between the component developers.
• Windows being largely dominant in the desktop/laptop market, a specific focus will be placed on the port of SAGE, and therefore all the components included in its distribution (in particular PARI/GP, GAP, SINGULAR, LINBOX) to this platform (D3.7).
• The deployment of a common infrastructure for multi-platform continuous integration (testing, building and distribution) will be addressed (D3.8).

T3.2 Interfaces between systems M0-M36
Sites: UPSud (lead), UVSQ, UOXF, USTAN, UGent

In this task we will investigate patterns to share data, ontologies, and semantics across computational systems, possibly connected remotely. We will leverage the well established semantics used in mathematics (categories, type systems, ...) to give powerful abstractions on computational objects. We will build upon the work already done in the EU FP6 project 26133 “SCIEnce – Symbolic Computation Infrastructure for Europe” (http://www.symbolic-computing.org/) on the Symbolic Computation Software Composability Protocol (SCSCP). SCSCP is a remote procedure call protocol by which a computer algebra system (CAS) may offer services to a variety of possible clients, including e.g. another CAS running on the same computer system or remotely; another instance of the same CAS (in a parallel computing context); a simplistic SCSCP client (e.g. C/C++/Python/etc. program) with a minimal SCSCP support needed for a particular application; a Web server which passes on the same services as Web services, etc. A distinctive feature of the protocol is that both instructions and data are represented in the OpenMath format (http://www.openmath.org/; previously supported by the EU JEM Thematic Network; EU project 24969 “ESPRIT” and other projects); moreover, OpenMath support is not limited by existing official OpenMath content dictionaries – private encodings may be easily embedded into SCSCP messages. SCSCP is already supported by a number of computer algebra systems, including GAP, Macaulay2, Maple, TRIP and others. We will extend support for SCSCP to other relevant systems involved in OpenDreamKit (D3.3). Through its API, we will enable discovery of subsystems, functionality, documentation and computational resources. The user interfaces shall be enabled to automatically choose the best available algorithms and resources to perform a required computation, as well as clearly and intuitively present the available choices to the expert user. As a first concrete test bed, we will consider the SAGE interface to GAP, or more precisely LIB GAP. Like most SAGE interfaces, this uses the now classical handle design pattern, whereby one can manipulate from SAGE an object created and stored in GAP, through a handle (a.k.a. remote objects). By mapping GAP’s categories to SAGE’s categories, in D3.9 we will:

- Implement a modular infrastructure for adapters, based on SCSCP, in order to let the implementation of adapters scale to a large variety of objects.
- Refactor the existing adapters, using this infrastructure to generalise their features. This step by itself will provide adapters for larger categories like semigroups or monoids.
- Merge the adapters into the handles, so that a handle to a GAP group will automatically behave like a native SAGE group.

A specific challenge will be performance; indeed low level method adapters, e.g. for arithmetic, need to be compiled when most of the interface infrastructure is dynamic by nature.

T3.3 Modularisation and packaging M0-M48
Sites: UVSQ (lead), UPSud, Logilab, UGent

In this task we will investigate best practices for composing, sharing and interfacing computational components and data for connected mathematical systems.

We will start with a comparative study of the practices adopted in various open source projects, both inside and outside of OpenDreamKit. This will include reviewing non-mathematical systems, e.g.: operating systems, platforms, web frameworks, cloud and HPC infrastructures. In particular, pushed by cloud computing, containerisation [6] and virtualisation [40] have become a major trend for distributing complex software, thanks to their ease of installation and configuration. We shall experiment with these technologies by building and distributing virtual appliances for the major components of OpenDreamKit (D3.1).

Once the initial study will have identified the present shortcomings, we will promote a new generation of mathematical software that is capable of scaling to large code bases, large datasets, and massively distributed infrastructures. This task also needs to consider the results of work package WP7 on social issues regarding distributed development, community management, acknowledging contributions, etc.

As an example, SAGE has a long history of integrating and distributing large mathematical libraries/software as a whole, with relatively few attention given to defining and exposing interfaces. Component re-usability is not a main focus for the SAGE community, at the same time the non-standard and relatively underused package system discourages writing and maintaining autonomous libraries. These factors have contributed to make the SAGE distribution what is usually described as a "monolith" (SAGE library code alone, not counting included libraries, makes up for 1.5M lines of code and documentation), hard to distribute, to maintain, to port, and to develop with. On the other hand, GAP has been distributing community-developed "GAP packages" for a long time, but faces now fragmentation issues, at the code and at the community level. The rudimentary package system adds more technical difficulties to GAP's development model.

Both models reach the limits of their scalability, and a synthesis is very much needed. Our first experiment will be to enhance SAGE's package system (D3.6), enough to support an open repository of user-contributed code, in the same spirit of modern systems such as Julia (http://pkg.julialang.org/) or PyPI (https://pypi.python.org/). Once internal packaging has been dealt with, the route will be paved to further modularise the SAGE distribution, and make sure that the major Linux distributions have standard packages for it (D3.10).

T3.4 Simulagora integration M0-M48
Sites: Logilab (lead)

To deliver every six month a new Simulagora VM image containing all the software components released over the period. The goal is to prove that the project is improving the component architecture by measuring the time it takes to integrate them.

T3.5 Component architecture for High Performance Computing and Parallelism M36-M48
Sites: UJF (lead), USTAN

As in all other areas of science, properly supporting massively parallel architecture is a major challenge. Many of the computational components have already gone a long way in this direction, and further work will be carried out in WorkPackage WP5. In this task we will investigate and implement parallelism-friendly ways of combining components together, so that calling components can benefit from the parallelism features of called components, with self-adaptation to the environment and cooperative sharing of resources. We will use SAGE and its components as a test-bed, by producing an HPC-enabled distribution

(D3.11).

T3.6 Document and modularise SageMathCloud's codebase M0-M24
Sites: UPSud (lead), UVSQ, UGent

From its inception in 2013, SageMathCloud (see Section 1.3.9, page 17) has quickly developed into a full featured VRE. Because of the tight, partly closed source development cycles, SAGE M ATH C LOUD's codebase has quickly grown in size, with its documentation not always keeping the pace. As a result, it is at the moment very hard for a newcomer to set up a clone service of SageMathCloud just from its sources. Now that SageMathCloud is fully open source, we need to go through its codebase, understand and document it (D3.2), isolate components that might be reused by other software (e.g.: J UPYTER), and make it as portable as possible. The ultimate goal of this task is to produce a personal version of SageMathCloud, to be shipped along with SAGE, that a user can run on his own personal computer (D3.4).

T3.7 Improving the development workflow in mathematical software M6-M24
Sites: UVSQ (lead), UPSud, Logilab, UGent

Truly open software must enable any actor to easily contribute his work to the community. Be it an experienced developer, or a student. Be it for a major software component or for a piece of translation. All the systems involved

in OpenDreamKit have developed their own workflows for contributing back, but those are almost exclusively geared toward experienced developers working on large components. When these workflows eventually reach their scalability limits, software development stagnates and major features are delayed. A well known example is given by SAGE's TRAC server, where tickets can stay in "needs review" state for a long time before entering the code base. Upstream bug reporting and fixing is another major factor of slow development. This task will seek new ways of accepting contributions to mathematical software in a scalable way. For example we will experiment with integrating bug reporting and contributing features right in the VRE (e.g., in SageMathCloud: D3.5).).

T3.8 Python interface for OOMMF micromagnetic simulation library M7-M13
Sites: SOUTHAMPTON (lead), USTAN

In this task, we create a Python interface for the open source Object Oriented MicroMagnetic Framework (OOMMF [1]). As a result, the OOMMF library will be fully accessible and usable from a Python interface and become a component in the Python/J UPYTER eco system of computational tools and in OpenDreamKit. We make use of this component architecture in T4.11 to build the micromagnetic VRE demonstrator (Sect. 1.3.7).

In more detail, we will first identify the best option for interfacing from Python to OOMMF core (C++) routines. The technical options include CTypes, Cython, Swig, and Boost-Python, all with particular advantages/disadvantages. Following analysis of the current OOMMF code layout and compilation model, we will use the most suitable tool, bearing in mind our ambition not to modify the OOMMF code so that the python interface we create remains functional and maintainable with minimal effort while the OOMMF core code is developed further by the OOMMF authors. The interface will expose the C++ objects in Python, providing an architecture component that provides full access to OOMMF's raw capabilities. For clarity, we will refer to this interface as OOMMF-py-raw.

Secondly, we will create a user friendly Python library OOMMF-py that combines the OOMMF-py-raw capabilities in an object orientated and safe-to-use Python library targeting researchers in the magnetic materials community. We will follow the design of the well-received high level Python interface in the Nmag micromagnetic simulation package [9] interface [28]. Unit and regression tests for both component interfaces OOMMF-py-raw and OOMMF-py are simultanously developed.

Participation per Partner

Partner number and short name	WP3 effort
1 - UPSud	50.00
4 - UJF	6.00
6 - UOXF	4.00
9 - SOUTHAMPTON	6.00
10 - USTAN	16.00
11 - UVSQ	8.00
14 - Logilab	14.00
16 - UGent	14.00
Total	118.00

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D3.1	Virtual images and containers	11 - UVSQ	Other	Public	6
D3.2	Understand and document SAGEMATHCLOUD backend code.	1 - UPSud	Report	Public	18
D3.3	Support for the SCSCP interface protocol in all relevant components (SAGE, GAP, etc.) distribution	10 - USTAN	Other	Public	12
D3.4	Personal SAGEMATHCLOUD: single user version of SAGEMATHCLOUD distributed with SAGE.	1 - UPSud	Other	Public	24

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D3.5	Integration between SAGEMATHCLOUD and SAGE’s TRAC server	11 - UVSQ	Other	Public	24
D3.6	Open package repository for SAGE	11 - UVSQ	Other	Public	24
D3.7	One-click install SAGE distribution for Windows with Cygwin	1 - UPSud	Other	Public	24
D3.8	Continuous integration platform for multi-platform build/test.	11 - UVSQ	Demonstrator	Public	36
D3.9	Semantic-aware SAGE interface to GAP.	6 - UOXF	Other	Public	36
D3.10	Packaging for major Linux distributions	11 - UVSQ	Other	Public	48
D3.11	HPC enabled SAGE distribution	4 - UJF	Other	Public	48

Description of deliverables

D3.1 : Virtual images and containers [6]
Virtual images and containers

D3.2 : Understand and document SAGEMATHCLOUD backend code. [18]
Understand and document SAGEMATHCLOUD backend code.

D3.3 : Support for the SCSCP interface protocol in all relevant components (SAGE, GAP, etc.) distribution [12]
Support for the SCSCP interface protocol in all relevant components (SAGE, GAP, etc.) distribution

D3.4 : Personal SAGEMATHCLOUD: single user version of SAGEMATHCLOUD distributed with SAGE. [24]
Personal SAGEMATHCLOUD: single user version of SAGEMATHCLOUD distributed with SAGE.

D3.5 : Integration between SAGEMATHCLOUD and SAGE’s TRAC server [24]
Integration between SAGEMATHCLOUD and SAGE’s TRAC server

D3.6 : Open package repository for SAGE [24]
Open package repository for SAGE

D3.7 : One-click install SAGE distribution for Windows with Cygwin [24]
One-click install SAGE distribution for Windows with Cygwin 32bits and 64bits

D3.8 : Continuous integration platform for multi-platform build/test. [36]
Continuous integration platform for multi-platform build/test.

D3.9 : Semantic-aware SAGE interface to GAP. [36]

Semantic-aware SAGE interface to GAP.

D3.10 : Packaging for major Linux distributions [48]
Packaging for major Linux distributions

D3.11 : HPC enabled SAGE distribution [48]
HPC enabled SAGE distribution

Schedule of relevant Milestones

Milestone number [18]	Milestone title	Lead beneficiary	Due Date (in months)	Means of verification
MS1	Startup	1 - UPSud	12	By Milestone 1 we will have carried out the requirements study, design and prototype implementations and started community building activities.
MS2	Implementations	1 - UPSud	24	By Milestone 2 we will have constructed first fully functional interface implementations and released enhanced versions of OpenDreamKit components, and train early adopters of Open-DreamKit.
MS3	Community/ Experiments	1 - UPSud	36	By Milestone 3 we will have gathered and evaluated feedback on OpenDreamKit software and established the portfolio of experiments produced with OpenDreamKit through further engaging with the community.
MS4	Evaluation	1 - UPSud	48	By Milestone 4 we will have released final versions of all OpenDreamKit components and completed the project evaluation.

Work package number [9]	WP4	Lead beneficiary [10]	15 - Simula
Work package title	User Interfaces		
Start month	1	**End month**	36

Objectives

The objective of this work package is to provide modern, robust, and
flexible user interfaces for computation, supporting real-time
sharing, integration with collaborative problem-solving, multilingual
documents, paper writing and publication, links to databases, etc.

Description of work and role of partners

WP4 - User Interfaces [Months: 1-36]
Simula, UPSud, CNRS, JacobsUni, UNIKL, USlaski, USFD, SOUTHAMPTON, USTAN, UVSQ, Logilab, UGent
Project JUPYTER (formerly IPYTHON notebook) provides a browser based
approach to constructing executable documents which comprise of code
(in multiple languages), mathematics, text, diagrams (see Section
1.3.4). The framework is an ideal portal through which VRE can be
operated. In this work package, we will add new functionality to the J
UPYTER notebook that fosters excellence in computational science and
research. In particular, we will push towards reproducible and
effective science by allowing structured documents (such as reports,
books, theses) from notebooks, and by allowing those notebooks to be
re-executed as self-contained regression tests. We will unify the
notebook infrastructure used in SAGE with JUPYTER, push forward
dynamic documentation exploration capabilities, and work towards
concurrent multi-user editing of notebooks. We will also develop
exemplar JUPYTER notebooks for education and research (e.g. T2.9).

To demonstrate the power of the OpenDreamKit environment to accelerate
computational science, deliver better value for money and make
computational science more robust, we will put together a
micromagnetic VRE (1.3.7) as a demonstrator.

T4.1 Uniform notebook interface for all interactive components M0-M36
Sites: UPSud (lead), Simula, UNIKL, USFD, SOUTHAMPTON, Logilab, USTAN, UVSQ, UGent

In this task, we will implement JUPYTER interfaces for the
interactive computation components of OpenDreamKit, including GAP,
PARI/GP, SAGE, and Singular. A first release D4.4 will focus on basic
functionality, and a second release D4.7 will cover advanced features
like 3D graphics or transparent documentation browsing (as live
worksheets whenever relevant). One of our objectives is to ensure the
sustainability of the project (Objective 5). The current SAGE
notebook interface was developed alongside that of JUPYTER, but with
slightly different goals. A notebook interface for SAGE is a vital
integrative component, and development was fast tracked to ensure its
availability to allow the project to move forward. However, JUPYTER,
whilst it initially proceeded more slowly, has a larger developer base
and has now caught up with the SAGE notebook in terms of
functionality. In line with Objective 5 SAGE will now phase out its
own notebook and switch focus to the JUPYTER notebook, outsourcing
this key but non disciplinary component.

The SAGE and JUPYTER convergence D4.5 will require:
• Robust migration path and tools for SAGE worksheets,
• Support for math, 2D, and interactive 3D scene visualisation,

• Import and export of ReST documents, with full support for SAGE's specific roles (math, ...),
• Support for remote SAGE kernel, typically on the cloud, or running with a different Python version (SAGE as a library),
• A migration path for interactive widgets implemented with SAGE's @interact functionality.

Joint meetings and visits between the developers of JUPYTER and of the computing components will be a key component of this task.

T4.2 Notebook improvements for collaboration M0-M24
Sites: Simula (lead), UPSud, USFD, JacobsUni, SOUTHAMPTON, Logilab

In this task, we will further improve tools for collaboration between authors of shared JUPYTER notebooks and draw from the experience of collaboration as set in Simulagora, SageMathCloud, etc. Version control tools, such as Git and Mercurial, have become an integral part of open and collaborative science and software. Version control tools allow proposed changes to be reviewed ('diffing') and resolve conflicts through combination of changes ('merging'). JUPYTER notebook documents are stored in text files as JSON formatted data. This makes them well suited to tracking in version control, but the JSON structure can make diffing and merging difficult. We will deploy tools to provide better support for visual diffing and merging of Notebook documents. These tools will be integrated into existing version control workflows D4.6. The MathHub.info system already has a distributed Git-based versioning system, which can serve as an entry point here.

Given the interactive nature of JUPYTER notebooks, live collaboration, where multiple authors work on the document simultaneously (like in Google Docs) is particularly desirable. However, there are particular challenges for collaborative editing of executable documents. The potential for shared execution adds both value and challenge to the live collaboration. Some attempts have been made to deal with live collaborative sessions (e.g. SageMathCloud, Colaboratory) but so far these have been outside the core JUPYTER project. In this task we will explore different models of single-notebook collaboration, including shared or separate execution D4.6. We will consider not only indicating authorship, but which author triggered which execution, and explore other challenges. Various avenues for live session collaboration will be explored for integration into JUPYTER itself D4.15.

T4.3 Reproducible Notebooks M12-M24
Sites: Simula (lead), UPSud, SOUTHAMPTON

In this task, we will develop tools that allow re-execution notebook documents with automated regression testing. The computed output will be compared against the stored output, and deviations reported as assertion errors. Notebooks are used in a variety of contexts, like training and teaching material (tutorials, documentation, books) or computer experimentation logbooks, where reproducibility is critical. Reproducibility dictates that the notebooks should remain functional and correct in the long run, even when the underlying computational software or infrastructure changes over time or across platforms.

This task is a critical component of reproducibility, allowing the notebook author to get an immediate notice when, e.g., a backward incompatible change occurs. It becomes even possible to anticipate such situations upstream by including important notebooks directly in

the automated test suite of the computational software, giving an easy
way for casual users to contribute regression tests.

Technically speaking, JUPYTER notebooks store outputs as rich
mime-type structures, with JSON metadata. Using this metadata, it will
be possible to express expectations of output, allowing more flexible
and powerful tests than direct text comparison D4.9. Prior work has
been done in SAGE for ReST files, e.g. sage −t notebook.rst, and this
model will be extended to notebooks.

T4.4 Refactor SAGE's S PHINX documentation system M0-M36
Sites: UPSud (lead), Simula, UVSQ, UGent

SAGE, like PYTHON and many other PYTHON based projects, uses the
SPHINX documentation system. Due to particularly stringent needs, many
layers of customisation and adaptations have accumulated over the
years, in particular for proper scaling to the sheer size of the Sage
documentation (13k pages just for the reference manual). A deep
refactorisation (D4.13) is critically needed to get rid of multiple
duplication, and foster sustainability by outsourcing back to SPHINX
all generic aspects (parallel compilation, index generation, ...).

T4.5 Dynamic documentation and exploration system M0-M12
Sites: UPSud (lead), Simula, SOUTHAMPTON, UVSQ, Logilab, UGent

Introspection has become a critical tool in interactive computation,
allowing user to explore, on the fly, the properties and capabilities
of the objects under manipulation. This challenge becomes particularly
acute in systems like SAGE where large parts of the class hierarchy is
built dynamically, and static documentation builders like S PHINX
cannot anymore render all the available information.

In this task, we will investigate how to further enhance the user
experience. This will include:
• On the fly generation of Javadoc style documentation, through
introspection, allowing e.g. the exploration of the class hierarchy,
available methods, etc.
• Widgets based on the HTML5 and web component standards to display
graphical views of the results of SPARQL queries, as well as
populating data structures with the results of such queries,
• D4.16 (Month 36) Exploratory support for semantic-aware interactive
widgets providing views on objects of the underlying computational
or databa se components. Preliminary steps are demonstrated in the
Larch Environment project (see demo video on
http://www.larchenvironment.com/) and SAGE-EXPLORER (https://github.
com/jbandlow/sage-explorer). The ultimate aim would be to
automatically generate LMFDB-style interfaces. Whenever possible,
those features will be implemented generically for any computation
kernel by extending the JUPYTER protocol with introspection and
documentation queries.

T4.6 Structured documents M0-M24
Sites: JacobsUni (lead), Simula, USFD, Logilab
JUPYTER notebooks consist of a sequence of cells that contain either
text or a program (see Section 1.3.4). Complex documents, such as
books, articles or reports, require a richer description that covers
the the structure of the document and the semantics of its
elements. This task will investigate this problem and try to find a
way to write these documents exploiting the breakthroughs achieved in
the other tasks to this workpackage. Several technical complementary
options can be explored:
• MathHub.info is a portal for reading and interacting with "active

documents” (i.e. documents that have an additional semantic layer
that supports semantic services like definition lookup,
type-inference, unit conversion,...)
• JUPYTER notebooks are essentially “programs with documentation” and
lack the semantical structure needed by complex documents.
• sTeX is a semantic variant of LaTeX that can be transformed into
OMDoc/MMT, which is the native knowledge representation format for
active documents and machine-actionable knowledge about math and
symbolic programs.

After gathering the needs and the requirements for the writing of
complex documents in the mathematical field, we will study these
design and build a solution that meets the expectations (D4.2). The
implementation will be achieved through an iterative process that
incrementally improves existing software solutions, making them
interoperable and synergistic. Results of this convergence will be
reported in D4.8, D4.5 and D4.11 and used in T2.9.

T4.7 Active Documents Portal M12-M36@.5
Sites: JacobsUni (lead)

We will extend the existing http://mathhub.info system to a portal for
interacting with active/structured documents (see T4.6) and releasing
the portal initially for internal use in the OpenDreamKit and later
for general use. MathHub. info already provides very basic sTeX
editing and versioning. In OpenDreamKit we will extend it on the
computational side based on the integrated format from T4.6. The
resulting portal will be made available to the consortium as D4.3 and
would be used for semantically enhanced code documentation and
knowledge representation (see WP6).

T4.8 Visualisation system for 3D data in web-notebook SOUTHAMPTON M0-M24
Sites: Simula (lead), USlaski, UPSud,

The JUPYTER notebook provides an attractive environment for building
user interfaces for research. However, the current support for inline
visualisation is limited to curve plots and 2D scalar fields. Many
scientific simulations need visualisation of 3D scalar and vector
fields, as shown in Figure 1.3.4. Experimentations in low dimensional
topology and differential geometry also relies on good drawing
capabilities (e.g. SnapPy or SageManifolds based on IPYTHON and
SAGE). The amount of data can be tremendous, especially in
time-dependent problems computed in a distributed fashion over
large-scale computational clusters. Interactive inspection of such
simulations can be a valuable tool which accelerates
research. However, for inspection, one does not need to transfer and
gather the full dataset at each time step—getting selected computed
fields on user request or preprocessing certain quantities like cross
sections with some predefined frequency will mostly suffice.

In this task we will first investigate available technologies for fast
in-browser visualisation of the typical structures to be displayed
(isosurfaces, streamlines, vector fields, cross sections, etc.). There
are several existing solutions which could provide basis for further
development. One of the best known, and also advanced is three.js
which provides basis for 3D visualisation in a web browser. Three.js
is WebGL based, but also provides canvas based rendering for system
which do not support WebGL. It has already been experimentally
deployed in Sage Cell Server and SMC projects. Another promising
technologies are visualisation libraries using exclusively
OpenGL. They can be deployed in browser based systems by using of the

WebGL API (which is a restricted subset of the regular OpenGL API). This can be accomplished by visualisation executed purely in GPU. VisPy and glumpy projects have found GPU-only solutions for common visualisation objects (lines, arrows, markers, text, iso-lines, iso-surfaces, text, etc) where data does not exit the GPU. The VisPy project already offers an experimental interface with Jupyter notebook that could be extended to cope with our specifications. Through this tight collaboration with the authors, OpenDreamKit could benefit from both dedicated and state-of-the art visualisation techniques.

The SnapPy and SageManifolds projects will be considered for deployment of tools we develop (see associated deliverable D4.12).

T4.9 Visualisation of 3D fluid dynamics data in web-notebook M12-M36
Sites: Simula (lead), USlaski, UPSud, SOUTHAMPTON

We propose to let computational fluid dynamics (CFD) be a driving application for the development of 3D visualisation in JUPYTER notebooks (T4.8) since CFD is one of the most demanding cases of scientific visualisation. The same time this task (with deliverable D4.14) will be demonstrator for (T4.8).

Successfully handling CFD makes the tool immediately applicable to a range of other fields such as heat transfer, electromagnetics, material science, and 3D algebraic structures in mathematics. Simulations would be initialised inside the notebook and executed on HPC clusters. This approach will significantly lower the threshold for using parallel computing codes that can be hard to install correctly on local workstations (see also WP5). Such use cases with 3D visualisation will greatly extend the potential applications of the JUPYTER notebook concept throughout science and engineering.

As example code for a 3D live web notebook with fluid dynamics simulations, we will use the Lattice Boltzmann solver which is under development at the University of Silesia: Sailfish. This code is an advanced Lattice Boltzmann solver designed from the ground up for distributed system of GPU compute clusters. It is implemented predominantly in Python, and it uses run-time code generation techniques to automatically build optimised code for CUDA and OpenCL devices. Since running Sailfish requires specialised hardware, it is reasonable to use it on dedicated HPC installations.

T4.10 Common option system for various displays in Sage M0-M24
Sites: CNRS (lead)

Given a mathematical object, it often has various possible representations on a computer. From raw text to LATEX, from simple 2d picture to a complicated 3d animation. In this task, we provide a uniform option system for displaying object within SAGE (raw text, LATEX, tikz, matplotlib, jmol, tachyon, ...). We will implement some of the missing display and will benefit of the work done in T4.9.

T4.11 Case study: micromagnetic VRE built from OpenDreamKit M13-M19
Sites: SOUTHAMPTON (lead), Simula,xUSFD

In this task, we use the OpenDreamKit architecture to assemble a virtual research environment software tailored for the large micromagnetic research community (see Section 1.3.7).

The micromagnetic VRE will be based on the JUPYTER notebook, the Python interface to the micromagnetic simulation library OOMMF (T3.8), and the additional features added to JUPYTER in this work package. The JUPYTER notebook environment allows to host, execute and document the Python-based OOMMF simulation in an executable document. In this interactive environment, objects can be displayed using various representations, including, for example, textual representation (i.e. strings), bitmap images and SVG (vector graphics) files. We will create functionality so that magnetisation vector field objects can be presented as a rendered 3d and 2d-view of the magnetisation field (Figure 1.3.4), and similar features for scalar fields such as field components and energies for static and time dependent data (linking to T4.9). This allows computational steering and the interactive exploration of the behavior of magnetic nanostructures.

Beyond that, the JUPYTER Widgets allow the creation of graphical user interface (GUI) elements, and we will generate code to display these widgets on demand to (i) set up micromagnetic simulations using a GUI, and (ii) assist in common post-processing simulation results. Recent pilot work has shown that it is possible to make JUPYTER Widgets interact with the Python interpreter session and this allows to activate a GUI-like (widget based) interface when desired but to quickly return to the interpreter prompt, taking forward the results (data) from the GUI session [29] and providing a continuous path from scripting to GUI. Having the ability to mix and match GUI-based and command driven analysis combines the best of both approaches, caters for users' preferences, and provides significant additional value.

T4.12 Python/Cython bindings for PARI M0-M48
Sites: CNRS (lead), UGent

PARI is a C-library and GP is its standalone interpreter. Partial Python/Cython bindings are provided by Sage. There is also a not more developed library CYPARI. The task aims to develop an independent Python/Cython library that would provide bindings for PARI/GP and which would tightly be developed within the PARI/GP team. Firstly, starting from SAGE and CYPARI, we will provide a standalone PARI bindings in Python and integrate it in SAGE (D4.1). Secondly, different optimisation will be provided for a tighter communication between PARI and PYTHON.
• Use the declaration files of GP to automatise CYTHON declarations.
• Replace copy from the PARI stack by direct pointers.
• More tests and documentation.
• Integrate the parallelisation features from T5.1 within PYTHON.
• Implement a crossed bug report system between SAGE and PARI (using results of T7.3).

The deliverable for this second step is D4.10. For this task we might require expertise of some SAGE, PARI or CYTHON developers (Jeroen Demeyer, Peter Bruin).

T4.13 Demonstrator: micromagnetic VRE notebooks M19-M25
Sites: SOUTHAMPTON (lead), Simula, UPSud

The purpose of the micromagnetic VRE (T4.11) is to enable excellent computational research. To maximise the value of this grant's investment for the community, we will not carry out micromagnetic research but instead produce a set of executable notebooks using the micromagnetic VRE to demonstrate its power and applicability. We will create executable notebook documents within the micromagnetic VRE including (i) a new tutorial on computational micromagnetics with OOMMF, (ii) the complete documentation of the OOMMF-Py library (T3.8), and (iii) a set of typical micromagnetic case studies. The tutorial, in terms of content, will take guidance from the tutorial provided for Nmag [27] and will introduce the additional features of the JUPYTER-driven micromagnetic VRE. We expect this substantial and executable documentation of the micromagnetic VRE to become the standard resource that introduces researchers to computational micromagnetics, in particular through the online portal (T4.14).

T4.14 Online portal for micromagnetic VRE demonstrator M25-M28
Sites: SOUTHAMPTON (lead), Simula, JacobsUni

Recently, a TeMPorary JUPYTER NoteBook (TMPNB) has been made available (at http://tmpnb.org) that allows anybody to open this URL and use their very own JUPYTER notebook for quick calculations and tests online. We will provide such a portal which provides the micromagnetic VRE for anonymous use. This service allows users to execute the demonstrator tutorial and documentation notebooks (T4.13) and run the calculations in real time on the web server, without having to install any software on their own machine. This web service will be based on Docker [6] virtualisation technology and we will make available the scripts to create VirtualBox [40] images, and Docker containers. The same virtual machine images can also be used for Cloud hosted computing services. We request e6000 to purchase a machine to provide these services (shared memory, 64 cores, 128GB RAM, Solidstate drive (SDD) to make the system more responsive).

Participation per Partner

Partner number and short name	**WP4 effort**
1 - UPSud	12.00
2 - CNRS	28.00
3 - JacobsUni	12.00
5 - UNIKL	2.00
7 - USlaski	4.00
8 - USFD	6.00
9 - SOUTHAMPTON	16.00
10 - USTAN	18.00
11 - UVSQ	2.00
14 - Logilab	12.00
15 - Simula	28.00
16 - UGent	14.00
Total	154.00

List of deliverables

Deliverable Number [14]	**Deliverable Title**	**Lead beneficiary**	**Type** [15]	**Dissemination level** [16]	**Due Date (in months)** [17]
D4.1	Python/Cython bindings for PARI and its integration in Sage	2 - CNRS	Other	Public	6

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D4.2	Active/Structured Documents Requirements and existing Solutions	3 - JacobsUni	Report	Public	9
D4.3	Distributed, Collaborative, Versioned Editing of Active Documents in MathHub.info	3 - JacobsUni	Demonstrator	Public	12
D4.4	Basic JUPYTER interface for GAP, PARI/GP, Singular	1 - UPSud	Other	Public	12
D4.5	SAGE notebook / JUPYTER notebook convergence	1 - UPSud	Demonstrator	Public	12
D4.6	Tools for collaborating on notebooks via version-control	15 - Simula	Other	Public	12
D4.7	Full JUPYTER interface for GAP, PARI/GP, SAGE, Singular	1 - UPSud	Other	Public	14
D4.8	Facilities for running notebooks as verification tests	15 - Simula	Other	Public	18
D4.9	In-place computation in active documents (context/ computation)	3 - JacobsUni	Demonstrator	Public	18
D4.10	Second version of the PARI Python/ Cython bindings	2 - CNRS	Other	Public	36
D4.11	Notebook Import into MathHub.info (interactive display)	3 - JacobsUni	Demonstrator	Public	24
D4.12	JUPYTER extension for 3D visualisation	15 - Simula	Other	Public	24
D4.13	Refactorisation of SAGE’s SPHINX documentation system	16 - UGent	Other	Public	24

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D4.14	Computational Fluid dynamics visualisation in web notebook	15 - Simula	Other	Public	36
D4.15	Exploratory support for live notebook collaboration	15 - Simula	Other	Public	36
D4.16	Exploratory support for semantic-aware interactive widgets providing views on objects represented and or in databases	1 - UPSud	Demonstrator	Public	36

Description of deliverables

D4.1 : Python/Cython bindings for PARI and its integration in Sage [6]
Python/Cython bindings for PARI and its integration in Sage

D4.2 : Active/Structured Documents Requirements and existing Solutions [9]
Active/Structured Documents Requirements and existing Solutions

D4.3 : Distributed, Collaborative, Versioned Editing of Active Documents in MathHub.info [12]
Distributed, Collaborative, Versioned Editing of Active Documents in MathHub.info

D4.4 : Basic JUPYTER interface for GAP, PARI/GP, Singular [12]
Basic JUPYTER interface for GAP, PARI/GP, Singular

D4.5 : SAGE notebook / JUPYTER notebook convergence [12]
SAGE notebook / JUPYTER notebook convergence

D4.6 : Tools for collaborating on notebooks via version-control [12]
Tools for collaborating on notebooks via version-control

D4.7 : Full JUPYTER interface for GAP, PARI/GP, SAGE, Singular [14]
Full JUPYTER interface for GAP, PARI/GP, SAGE, Singular

D4.8 : Facilities for running notebooks as verification tests [18]
Facilities for running notebooks as verification tests

D4.9 : In-place computation in active documents (context/computation) [18]
In-place computation in active documents (context/computation)

D4.10 : Second version of the PARI Python/Cython bindings [36]
Second version of the PARI Python/Cython bindings

D4.11 : Notebook Import into MathHub.info (interactive display) [24]
Notebook Import into MathHub.info (interactive display)

D4.12 : JUPYTER extension for 3D visualisation [24]
JUPYTER extension for 3D visualisation

D4.13 : Refactorisation of SAGE’s SPHINX documentation system [24]

Refactorisation of SAGE’s SPHINX documentation system

D4.14 : Computational Fluid dynamics visualisation in web notebook [36]
Computational Fluid dynamics visualisation in web notebook

D4.15 : Exploratory support for live notebook collaboration [36]
Exploratory support for live notebook collaboration

D4.16 : Exploratory support for semantic-aware interactive widgets providing views on objects represented and or in databases [36]
Exploratory support for semantic-aware interactive widgets providing views on objects represented and or in databases

Schedule of relevant Milestones

Milestone number [18]	**Milestone title**	**Lead beneficiary**	**Due Date (in months)**	**Means of verification**
MS1	Startup	1 - UPSud	12	By Milestone 1 we will have carried out the requirements study, design and prototype implementations and started community building activities.
MS2	Implementations	1 - UPSud	24	By Milestone 2 we will have constructed first fully functional interface implementations and released enhanced versions of OpenDreamKit components, and train early adopters of Open-DreamKit.
MS3	Community/ Experiments	1 - UPSud	36	By Milestone 3 we will have gathered and evaluated feedback on OpenDreamKit software and established the portfolio of experiments produced with OpenDreamKit through further engaging with the community.

Work package number [9]	WP5	**Lead beneficiary** [10]	4 - UJF
Work package title	High Performance Mathematical Computing		
Start month	1	**End month**	48

Objectives

The objective of this work package is to improve the performance of the computational components of OpenDreamKit, in particular on massively parallel architectures. This includes notably:

- Fine grained High Performance Computing on many-cores architectures.
- Coarse grained or embarrassingly parallel computing on grids or on the cloud.
- Compilation of high level interpreted code to optimised parallel native code.
- Develop novel HPC infrastructure in the context of combinatorics.

A key aspect will be to foster further sharing expertise and best practices between computational components.

Description of work and role of partners

WP5 - High Performance Mathematical Computing [Months: 1-48]
UJF, UPSud, CNRS, UNIKL, USFD, USTAN, Logilab
As in all other areas of science, properly supporting massively parallel architectures is a major challenge. Many of the computational components in OpenDreamKit have already gone a long way in this direction. For example, parallel versions of the GAP kernel for a range of architectures were developed during the 2009-2013 EPSRC "HPCGAP" project. The expertise gained there was then transferred to the ongoing S INGULAR-HPC project, in particular through the rehiring of one of the developers of HPC-GAP. The French ANR HPAC project (2012-2015) has also widely contributed to design parallel exact linear algebra kernels which is a core component for most HPC applications. The LinBox library, used in sage, has benefited from this experience on the multi-core processing aspects. In this work package, we will build on this momentum to further implement HPC support in the component Tasks T5.1 (PARI),T5.2 (GAP), T5.3 (LinBox), T5.5(MPIR) and T5.4 (S INGULAR).

Many of the computational components of OpenDreamKit, notably SAGE and GAP use a high level interpreted language for their library. Performance is achieved by rewriting or compiling critical sections into a lower-level language. SAGE uses the CYTHON PYTHON-to-C compiler; GAP has some more basic support. In Tasks T5.2 and T5.7, we will also boost performance by further developing and applying such compilation tools, allowing the application programmer to retain their high level approach.

T5.1 PARI M0-M48@0.5
Sites: CNRS (lead)

PARI is a C library mainly oriented toward arithmetic and number theory. It currently supports both POSIX threads or MPI but lacks interfaces for parallelism. More precisely, it should be easier from an external package or software (e.g. Sage) to better exploit PARI parallel features. On the other hand, most basic algorithms in the PARI library (e.g. integer factorisation) are currently implemented using only one core. To make better use of multi-core architecture and more generally parallel architectures, we will devise a generic parallelisation machinery to allow individual implementations to scale gracefully between single core / multicore / massively parallel machines while avoiding code duplication. The deliverables for this task are D5.10 and D5.16.

T5.2 GAP M0-M48@0.375
Sites: USTAN (lead)

Thanks to the HPCGAP project, almost the full functionality of GAP can be safely run on multicore architectures, and there is support for simple but effective parallel programming, with protection from most of the more serious pitfalls that can trouble the novice parallel programmer. Experimental versions of GAP also exist for a number of distributedmemory architectures. In this task we will continue to develop the GAP infrastructure to offer performance improvements to real end users on a wide range of modern hardware. This will be achieved by a number of synergistic developments in the system:
• a library of parallel algorithms for algebraic computation. This will include general purpose skeletons applicable to many problems; distributed data structures suitable for orchestrating distributed memory computations; and implementations of specific parallel algorithms for key mathematical tasks. Target skeletons include irregular parallel maps and folds; transitive closure operations, especially orbits of group actions and chain reduction (as used in Gaussian elimination). Data structure targets include distributed task queue, hash table and array structures.Specific algorithm targets include linear algebra over finite fields; randomised search algorithms in general and matrix group recognition in particular and algorithms for analysing the structure of groups given by a finite presentation
• interfaces between GAP and standard cloud and HPC infrastructure. This work will be based on T3.5. At the moment GAP is designed for interactive use or use as a local (lacking resource control or security) SCSCP server. Data is taken from local files or fixed URLs. We will develop interfaces that make it easy to run GAP jobs through standard batch queue environments; enable SCSCP servers to take advantage of widely available authentication and resource control frameworks in cloud and HPC settings, and access resources through standard discovery and allocation mechanisms.

• adaptation of the cython and/or pythran technology to allow the performance of critical GAP language subroutines to be increased close to that of C code without the programmer cost of a full C implementation. The deliverables for this task is D5.15, a report on all the GAP-related activities of this workpackage, including links to the software which will by then have been publically released.

T5.3 Linbox M3-M48@0.37; M12-M36
Sites: UJF (lead)

Most intensive mathematical computations rely heavily on exact linear kernels and their ability to harness parallel computers, grids or clusters. The LinBox library, already delivers high sequential efficiency for mathematical software such as SAGE. The parallelisation of the library for multi-core architectures has been initiated in the A.N.R. HPAC project and successfully set the building blocks for high performance algebraic computing. The task here is to address the remaining challenges for the use of such kernels through a general audience mathematical software, such as SAGE. A first aspect focuses on code design and domain specific languages allowing to expose an abstraction of the parallel infrastructure and the parallel features of the code through the stack of libraries, and support the composition of parallel routines. This will be addressed in deliverable D5.9. More generally the second aspect, addressed in deliverable D5.12 concerns the development of new parallel algorithms and implementations, that are still barriers in the development of High performance mathematical applications. Lastly, the third part, in deliverable D5.14, addresses the specificities of distributed computing, with a close focus on communications and heterogeneous infrastructures.

T5.4 Singular M0-M48@0.9
Sites: UNIKL (lead)

The unique challenge of parallelizing Singular has been that it is a decades-old project, with a codebase exceeding 300,000 lines of code and an enormous existing investment of development effort. This makes a wholesale manual rewrite or reengineering approach infeasible. We therefore use a multi-pronged approach: First, we have created automated source-to-source translation tools that take existing C/C++ code as input and generate thread safe code as output. Secondly we are also adding facilities to the C/C++ code and the Singular interpreter to safely access shared memory. These facilities ensure in particular that common pitfalls of concurrent programming, such as data races and deadlocks, cannot occur. For this, we leverage approaches that have already been successfully used for HPC-GAP and whose principles are well-understood. To supplement the above existing work, we propose to add very fine-grained parallelism to some key components of Singular. These include writing a multi-threaded implementation of the Singular multivariate polynomial arithmetic, of the main quadratic sieve implementation for integer factoring and parallelisation of the FFT based integer and dense polynomial multiplication algorithm. These key components are used extensively for Singular’s overall workload, including in the Groebner basis engine and polynomial subsystems. Performance increases through fine-grained parallelisation of key components such as these will provide extensive benefits to virtually all users of Singular on multi-core machines. Output are D5.6, D5.7 and D5.13.

T5.5 MPIR M6-M18
Sites: UNIKL (lead)

MPIR (Multiple Precision Integers and Rationals) is the core library in SAGE for bignum arithmetic. It is used extensively by a majority of the core C/C++ libraries in SAGE, and by SAGE directly via CYTHON. MPIR is a fork of the GMP (Gnu Multi-precision) library, with many independent implementations of the core algorithms (including a faster FFT and division code, better superoptimisation on some common 64 bit processors and native MSVC support). It consists of around 250,000 lines of code, much of which is assembly primitives and very low level, highly optimised C code. Maintenance of MPIR is not merely a matter of updating the build system. Rather, every time a new processor is released by AMD, Intel, Sparc or ARM, significant development has to be invested in hand-optimising and then superoptimising assembly code for the new processors. This gives up to a 12x performance increase over optimised C code, due to the specialised nature of bignum arithmetic, which is in some sense a worst case for C compilers. Indeed without continuous effort, MPIR would not even run on new operating systems and processors, let alone run fast. This is a unique problem that assembly libraries have.

As a successful and key component of SAGE, we believe it is time to invest in maintenance and improvement of MPIR by hiring an assembly expert who can work full time on the project after MPIR's lead assembly expert sadly passed away recently. Significant challenges exist, such as optimising for SIMD instruction sets. Without investment into maintenance, assembly superoptimisation, processor support, fat binary support, etc. this key component of SAGE will fall behind, to the detriment of SAGE as a whole and the numerous other standalone libraries that make use of MPIR. Output is D5.5 and a contribution to D5.7.

T5.6 HPC infrastructure for combinatorics M0-M6@0.3; M12-M36@0.5
Sites: UPSud (lead), CNRS

Several members of the projects are experts in combinatorics a field where Sage is clearly a world leader [4]. This particular research field has several specific features which makes it interesting from the HPC point of view. The most important feature is that the goal of research is mostly to design and to understand properties of algorithms. As a consequence, much more often that in other field, the researcher needs to program. However, this is not his ultimate goal so the programming environment must be very expressive for fast prototyping. At the same time, the problems often require relatively large computations; algorithms of exponential complexity are extremely common, and combinatorial explosion is the main obstacle to many experiments. Hence the programming environment must make no compromise on efficiency.

Finally, embarrassingly parallel problem are extremely common, and more often than not problems that are not embarrassingly parallel reduce to the exploration of a large tree. Hence the programming environment must minimise the extra work needed to get from a serial program to a parallel one in these simple situations. Through this task we will provide a concrete, practical, and highly demanding use cases for the infrastructure developed in this work package. In particular, they will serve to evaluate the benefits of tasks T5.7, T3.3, and T3.5. In particular, we will provide a mixed C/Python implementation that will be integrated within Sage and replace most of the Sage-combinat code (deliverable D5.1 and D5.11). In a second and more exploratory direction, some experiments [10] shows that the large tree exploration problem is very easily solved in C++ using the new Intel Cilk++ [35, 36] technology (See for example: https://github.com/jfromentin/nsgtree and https://github.com/hivert/IVMPG). We would like to explore the possibility to interface smoothly PYTHRAN, CYTHON, and Cilk++ (deliverable D5.8).

T5.7 Pythran M0-M24
Sites: Logilab (lead), UJF

CYTHON (a fork of Pyrex) is an extended-PYTHON to C compiler that has received significant contributions from SAGE developers, and is a thriving project of its own. PYTHRAN and CYTHON are similar in spirit but have complementary feature sets: PYTHRAN can heavily optimise high level N UMPY constructs and CYTHON has broader PYTHON support. In this task, we will investigate the opportunity and feasibility of a convergence between CYTHON and PYTHRAN.

• depending on the code at hand, one strategy or the other would be automatically selected.
• The optimised runtime of PYTHRAN can be used through CYTHON.

An effort will be made to improve more and more the parallelism in the PYTHRAN runtime. This work will be achieved through a close

collaboration between the PYTHRAN developers hired for OpenDreamKit
and CYTHON developers involved in the SAGE project. It should quicken
SAGE execution time at least on N UMPY centric codes, while not
putting an extra burden on the developers. Preliminary discussions
with the CYTHON community have already taken place and received a very
favorable feedback.

Adding PYTHRAN support in SAGE will be done not only for SAGE code but
also for SAGE users code thanks to compilation facilities in the
notebook interface. Output is D5.2.

Internally, SAGE uses CYTHON for compiling the critical sections of
its libraries. In this task, we will explore opportunities to benefit
from PYTHRAN compilation within the SAGE library to benefit PYTHRAN
compile time optimisation. A specific challenge is that the SAGE
library uses quite heavily object-oriented programming. A first step
to support object-oriented programming will be to make PYTHRAN type
inference more accurate, which will also improve error feedback
provided for the user. Output is D5.4.

T5.8 Sun Grid Engine Integration in Project Jupyter Hub M0-M12
Sites: USFD (lead)

The Sun Grid Engine is a commonly used High Performance Computing
scheduler. It is used, for example, on the institutional HPC systems
in both Sheffield and Manchester Universities as well as the regional
N8 HPC facility, a system shared by the 8 most research intensive
universities in the North of England. In this task, we will develop
and demonstrate a Sun Grid Engine notebook spawner for Project
Jupyter, allowing users to access Jupyter notebooks on the HPC
cluster. This will enable the interactive analysis of output data
products on the cluster where they were generated and are stored, via
a user-friendly web interface D5.3.

Participation per Partner

Partner number and short name	**WP5 effort**
1 - UPSud	6.00
2 - CNRS	40.00
4 - UJF	52.00
5 - UNIKL	60.00
8 - USFD	12.00
10 - USTAN	18.00
14 - Logilab	12.00
Total	200.00

List of deliverables

Deliverable Number [14]	**Deliverable Title**	**Lead beneficiary**	**Type** [15]	**Dissemination level** [16]	**Due Date (in months)** [17]
D5.1	Turn the Python prototypes for tree	1 - UPSud	Demonstrator	Public	3

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
	exploration into production code, integrate to SAGE.				
D5.2	Facility to compile PYTHRAN compliant user kernels and Sage code and automatically take advantage of multi-cores and SIMD instruction units in CYTHON	4 - UJF	Demonstrator	Public	18
D5.3	Sun Grid Engine support for Project Jupyter Hub	8 - USFD	Other	Public	12
D5.4	Make PYTHRAN typing better to improve error information.	14 - Logilab	Demonstrator	Public	12
D5.5	Extend the existing assembly superoptimiser for AVX and upcoming Intel processor extensions for the MPIR library.	5 - UNIKL	Demonstrator	Public	18
D5.6	Parallelise the relation sieving component of the Quadratic Sieve and implement a parallel version of Block-Wiederman linear algebra over GF2 and the triple large prime variant.	5 - UNIKL	Demonstrator	Public	18
D5.7	Take advantage of multiple cores in the matrix Fourier Algorithm component of the FFT for integer and polynomial arithmetic,and include assembly primitives for SIMD processor instructions (AVX,	5 - UNIKL	Demonstrator	Public	18

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
	Knight's Bridge, etc.), especially in the FFT butterflies				
D5.8	Explore the possibility to interface smoothly PYTHRAN, CYTHON and Cilk ++	1 - UPSud	Demonstrator	Public	24
D5.9	Library design and domain specific language exposing LINBOX parallel features to SAGE	4 - UJF	Report	Public	24
D5.10	Devise a generic parallelisation engine for PARI and use it to prototype selected functions (integer factorisation, discrete logarithm, modular polynomials)	2 - CNRS	Demonstrator	Public	24
D5.11	Refactor and Optimise the existing combinatorics SAGE code using the new developed PYTHRAN and CYTHON features	2 - CNRS	Demonstrator	Public	36
D5.12	Exact linear algebra algorithms and implementations. Library maintenance and close integration in mathematical software for LINBOX library	4 - UJF	Demonstrator	Public	36
D5.13	Parallelise the Singular sparse polynomial multiplication algorithms and provide parallel versions of the Singular sparse	5 - UNIKL	Demonstrator	Public	48

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
	polynomial division and GCD algorithms.				
D5.14	Implementations of exact linear algebra algorithms on distributed memory et heterogenous architectures: clusters and accelerators. Solving large linear systems over the rationals is the target application.	4 - UJF	Demonstrator	Public	48
D5.15	Final report and evaluation of the GAP developments	10 - USTAN	Report	Public	48
D5.16	PARI suite release (LIBPARI, GP and GP2C) that fully support parallelisation allowing individual implementations to scale gracefully between single core / multicore / massively parallel machines.	2 - CNRS	Demonstrator	Public	48

Description of deliverables

D5.1 : Turn the Python prototypes for tree exploration into production code, integrate to SAGE. [3]

Turn the Python prototypes for tree exploration into production code, integrate to SAGE.

D5.2 : Facility to compile PYTHRAN compliant user kernels and Sage code and automatically take advantage of multi-cores and SIMD instruction units in CYTHON [18]

Facility to compile PYTHRAN compliant user kernels and Sage code and automatically take advantage of multi-cores and SIMD instruction units in CYTHON

D5.3 : Sun Grid Engine support for Project Jupyter Hub [12]

Sun Grid Engine support for Project Jupyter Hub

D5.4 : Make PYTHRAN typing better to improve error information. [12]

Make PYTHRAN typing better to improve error information.

D5.5 : Extend the existing assembly superoptimiser for AVX and upcoming Intel processor extensions for the MPIR library. [18]

Extend the existing assembly superoptimiser for AVX and upcoming Intel processor extensions for the MPIR library.

D5.6 : Parallelise the relation sieving component of the Quadratic Sieve and implement a parallel version of Block-Wiederman linear algebra over GF2 and the triple large prime variant. [18]

Parallelise the relation sieving component of the Quadratic Sieve and implement a parallel version of Block-Wiederman linear algebra over GF2 and the triple large prime variant.

D5.7 : Take advantage of multiple cores in the matrix Fourier Algorithm component of the FFT for integer and polynomial arithmetic,and include assembly primitives for SIMD processor instructions (AVX, Knight's Bridge, etc.), especially in the FFT butterflies [18]

Take advantage of multiple cores in the matrix Fourier Algorithm component of the FFT for integer and polynomial arithmetic,and include assembly primitives for SIMD processor instructions (AVX, Knight's Bridge, etc.), especially in the FFT butterflies

D5.8 : Explore the possibility to interface smoothly PYTHRAN, CYTHON and Cilk++ [24]

Explore the possibility to interface smoothly PYTHRAN, CYTHON and Cilk++

D5.9 : Library design and domain specific language exposing LINBOX parallel features to SAGE [24]

Library design and domain specific language exposing LINBOX parallel features to SAGE

D5.10 : Devise a generic parallelisation engine for PARI and use it to prototype selected functions (integer factorisation, discrete logarithm, modular polynomials) [24]

Devise a generic parallelisation engine for PARI and use it to prototype selected functions (integer factorisation, discrete logarithm, modular polynomials)

D5.11 : Refactor and Optimise the existing combinatorics SAGE code using the new developed PYTHRAN and CYTHON features [36]

Refactor and Optimise the existing combinatorics SAGE code using the new developed PYTHRAN and CYTHON features

D5.12 : Exact linear algebra algorithms and implementations. Library maintenance and close integration in mathematical software for LINBOX library [36]

Exact linear algebra algorithms and implementations. Library maintenance and close integration in mathematical software for LINBOX library

D5.13 : Parallelise the Singular sparse polynomial multiplication algorithms and provide parallel versions of the Singular sparse polynomial division and GCD algorithms. [48]

Parallelise the Singular sparse polynomial multiplication algorithms and provide parallel versions of the Singular sparse polynomial division and GCD algorithms.

D5.14 : Implementations of exact linear algebra algorithms on distributed memory et heterogenous architectures: clusters and accelerators. Solving large linear systems over the rationals is the target application. [48]

Implementations of exact linear algebra algorithms on distributed memory et heterogenous architectures: clusters and accelerators. Solving large linear systems over the rationals is the target application.

D5.15 : Final report and evaluation of the GAP developments [48]

Final report and evaluation of the GAP developments

D5.16 : PARI suite release (LIBPARI, GP and GP2C) that fully support parallelisation allowing individual implementations to scale gracefully between single core / multicore / massively parallel machines. [48]

PARI suite release (LIBPARI, GP and GP2C) that fully support parallelisation allowing individual implementations to scale gracefully between single core / multicore / massively parallel machines.

Schedule of relevant Milestones

Milestone number [18]	Milestone title	Lead beneficiary	Due Date (in months)	Means of verification
MS1	Startup	1 - UPSud	12	By Milestone 1 we will have carried out the requirements study, design and prototype implementations and

Schedule of relevant Milestones

Milestone number [18]	Milestone title	Lead beneficiary	Due Date (in months)	Means of verification
				started community building activities.
MS2	Implementations	1 - UPSud	24	By Milestone 2 we will have constructed first fully functional interface implementations and released enhanced versions of OpenDreamKit components, and train early adopters of Open-DreamKit.
MS3	Community/ Experiments	1 - UPSud	36	By Milestone 3 we will have gathered and evaluated feedback on OpenDreamKit software and established the portfolio of experiments produced with OpenDreamKit through further engaging with the community.
MS4	Evaluation	1 - UPSud	48	By Milestone 4 we will have released final versions of all OpenDreamKit components and completed the project evaluation.

Work package number [9]	WP6	**Lead beneficiary** [10]	3 - JacobsUni
Work package title	Data/Knowledge/Software-Bases		
Start month	1	**End month**	48

Objectives

The ultimate purpose of a mathematical VRE is to create data (D; see Section 1.3.1), knowledge (K), and software (S) by modeling world situations, computing mathematical objects, and running computational experiments. To be effective a VRE needs an infrastructure that supports the creation, management, access, and dissemination of DKS-Structures. All the systems considered in this proposal (GAP, SAGE, PARI, SINGULAR, OEIS, arXiv.org, . . .) already include data, knowledge, and software modules as part of their regular distribution, but not in a form that is interoperable between systems, severely limiting the usefulness of the systems and results. The objectives of this work package are :
1. to design metadata and representation formats for trans-system DKS structures as a basis for a math VRE,
2. implement interfaces to existing systems for interoperability and compatibility with the RE, and
3. implement a joint DKS infrastructure for, searching, documentation, traceability, versioning, provenance, visualisation and native dissemination of OpenDreamKit results (the latter three together with WP4).
Concretely we will design and build an infrastructure that would make it easy for either individual mathematicians or a distributed collaboration to manage and use such interlinked mathematical data. This work would provide part of the backend to WP4, and would draw on previous work with the LMFDB and FINDSTAT (which will be treated as prototypes for our purposes, to serve as exemplars to other projects) and in return will substantially enhance their capabilities.
User prerequisites should be kept to a minimum (depending on contributors' and users' needs and goals), and in particular would not require any background in databases to contribute new data or perform queries.

Description of work and role of partners

WP6 - Data/Knowledge/Software-Bases [Months: 1-48]
JacobsUni, UPSud, USTAN, UWarwick, UZH, Logilab
We need ways to represent DKS in the same representational system, make the DKS structures explicit and therefore machine-manageable and – since current computational/experimental mathematics involve quite extensive DKS – we need a new kind of "database", which we will call Mathematical Data/Knowledge/Software-base (DKS base), and which we will build in this work package.

The starting points for this unification effort will be the system-oriented data bases for D , the OMDoc (Open Mathematical Documents) framework [17] for K. OMDoc/MMT [32] is a representation format for mathematical documents and knowledge that incorporates a metalogical framework to be foundation-independent, which allows interoperability between various ontologies/foundations of mathematics. For the integration of K and S we will build on the notion of biform theories developed by Carette/Farmer [8] and extended to OMDoc/MMT by JacobsUni in [19]. In this setup, the programming language serves as a foundation, just as ZFC set theory might for mathematical knowledge. Complex relationships between mathematical objects, interpretations of the underlying languages, and unit transformations are modeled in a graph of theories and theory morphisms.

The complexity of mathematical DKS structures is on vivid display in the L-functions and Modular Forms database project (LMFDB): while the general shape of the functional equation of an L-function is dependent on a lot of theoretical knowledge, it also requires parameter data and the coefficients of the associated Dirichlet series. Once this is obtained, highly optimised (and heavily parallelizable) algorithms can be run to compute values of this function.

T6.1 Survey of existing DKS bases, Formulation of requirements M0-M3
Sites: UZH (lead), JacobsUni, USTAN, UWarwick, USlaski

In this task, we will survey existing databases, the technology used to implement them, how they were linked to the rest of the existing infrastructure and the functionalities offered. We will also select additional external data and projects to add to this effort, aiming to maximise the impact of our work.

We will organise a workshop associated to this task (see T2.3). Results will be communicated in D6.2.

T6.2 Triform Theories in OMDoc/MMT M0-M12
Sites: JacobsUni (lead), UZH

Work here would extend OMDoc/MMT biform theories along the data axis, which will require a specialised but integrated treatment. This integration will serve as a theoretical basis informing the design of a DKS base in T6.3. The results are reported in D6.3.

T6.3 DKS Base Design M6-M12; M15-M18@.33
Sites: JacobsUni (lead), UZH, USlaski, USTAN, UWarwick, Logilab

Ontologies are the canonical method used to implement databases that require significant data interchange. However, because of the extreme reification present in mathematics (relations between objects themselves become objects of study), there are specific obstacles compared to the usual semantic web model of publishing. Drawing on semantic web/Linked Open Data experience of the Logilab group, specialised to mathematics through the OMDoc/MMT work of the Bremen group, we will design a decentralised infrastructure for OpenDreamKit. This infrastructure would allow modular collaborations, through decentralised hosting of data without the need to merge everything centrally.

The initial design of the DKS base in OpenDreamKit is reported in D6.2. Conversion issues are covered in T6.4.

T6.4 Computational Foundation for Python/Sage M6-M18@.66
Sites: JacobsUni (lead), UZH, USTAN

In the OMDoc/MMT world a foundation is a logical base language that gives the formal meaning to all objects represented/formalised in it. We have created a very initial computational foundation for the programming language Scala and implemented it in the MMT API. This can be used to execute (or verify) computations directly in OMDoc/MMT and thus forms the basis for various integration tasks for OMDoc/MMT biform theories that integrate Scala computations. Here we propose to develop a somewhat more complete computational foundation for Python and/or parts of Sage (coverage to be determined). Bi/Triform theories come in three parts:
• syntax: what operators/types are there, how do they nest,
• computation: what does the computation relation look like (sometimes called operational semantics). The declarative semantics of a computational foundation can be given as an OMDoc/MMT theory morphism into another foundation (e.g. a set theory);
• specification: what are the observable properties of the computation.
The foundation (a triform theory in OMDoc/MMT) will be published as D6.5.

T6.5 Knowledge-based code infrastructure M12-M48
Sites: UPSud (lead), UZH, JacobsUni

Over the last decades, computational components, and in particular Axiom, MuPAD, GAP, or SAGE, have embedded more and more mathematical

knowledge directly inside the code, as a way to better structure it
for expressiveness, flexibility, composability, documentation, and
robustness. In this task we will review the various approaches taken
in these software (e.g. categories and dynamic class hierarchies) and
in proof assistants like Coq (e.g. static type systems), and compare
their respective strength and weaknesses on concrete case studies. We
will also explore whether paradigms offered by recent programming
languages like Julia or Scala could enable a better
implementation. Based on this we will suggest and experiment with
design improvements, and explore challenges such as the compilation,
verification, or interoperability of such code.

T6.6 OEIS Case Study (Coverage and automated Import) M12-M18
Sites: JacobsUni (lead)

In this case study we test the practical coverage of the trifunctional
modules, by transforming an existing, high-profile database (the
Online Sequence of Integer Sequences9) into OMDoc/MMT. The OEIS has
about 250 thousand sequences, with formulae, descriptions,
definitions, references, software, etc. in a structured text file (but
no standardised format for formulae and references), so we expect to
get 250 k theories. Having the OEIS in OMDoc/MMT form allows to do
Knowledge Management services (presentation, definition lookup,
formula search, ...) in M ATH H UB (see WP4). The OEIS is a good case
study, since the data is licensed under a Creative Commons license
which allows derived works. The large size will allow statistically
significant semantic cross-validation of the heuristic transformation
process and thus achieve a significant community resource.

The results of the import are reported in D6.4.

T6.7 FindStat Case Study (Triformal Theories) M18-M30@.5
Sites: JacobsUni (lead), UZH

In this task we would develop triformal theories for the F IND S TAT
project 10 to test the design from T6.4. Similarly to the previous
task, in this case study, we first develop a thorough OMDoc/MMT model,
which should only involve a handful of MMT theories (combinatorial
collections, maps, statistics,...), each with a few hundred
realisations. Together with WP4, this will again allow for easier
knowledge management services, and in particular improved search
services. This Task will be co-developed with T6.4, it will validate
the design of triformal theories and be iterated to test the design
changes. Results will be reported in D6.4.

T6.8 LMFDB Case Study (Triformal Theories) M12-M24@.25; M24-M48@.7 Sites: JacobsUni (lead), UZH, UWarwick

In this task we would develop triformal theories for an exemplary part
of the LMFDB project to test the design from T6.4. We will identify a
fragment of the LMFDB that we want to model and design the model (see
D6.4). Then we will perform cross-validation of the three model parts
against each other (essentially model-based testing of software and
inference; see D6.8). Once this has been successful for the chosen
fragment, we will try to semiautomatically extend the import and model
to the whole LMFDB to gain coverage and integrate it fully into the
DKS base. We expect that this will entail quite a lot of work in
refactoring the LMFDB proper, which will benefit the LMFDB community
independently of its use in OpenDreamKit.

Finally, we will pick an algorithm from the LMFDB and verify it
against its specification and the computational foundation developed

in T6.4; this is the final validation of the case study. The results are reported in D6.11.

T6.9 Memoisation and production of new data M24-M42@.6
Sites: USTAN (lead), USlaski, UPSud, UWarwick

Many CAS users run large and intensive computations, for which they want to collect the results while simultaneously working on software improvements. GAP retains computed attribute values of objects within a session; SAGE currently has a limited cached method. Neither offers storage that is persistent across sessions or supports publication of the result or sharing within a collaboration. We will use, extend and contribute back to, an appropriate established persistent memoisation infrastructure, such as python-joblib, redis-simple-cache or dogpile.cache, adding features needed for storage and use of results in mathematical research. We will design something that is simple to deploy and configure, and makes it easy to share results in a controlled manner, but provides enough assurance to enable the user to rely on the data, give proper credit to the original computation and rerun the computation if they want to. Results are reported in D6.9.

T6.10 Math Search Engine M3-M9@.3; M21-M42@.5
Sites: JacobsUni (lead)

The advantage of having a unified DKS base for a math VRE is that we can navigate the combined information space of all the underlying tools, systems and resources integrated into the VRE. The negative effect is that this aggravates the already serious problem of finding anything. Therefore we will adapt the existing MathWebSearch Engine (MWS [20, 25]) to the DKS base system. MWS consists of a web service that indexes mathematical documents (formula/text) and a web front-end that allows users to query the index. Formula queries are highly efficient (25µs/query) and can be combined with keyword/full text search queries. An initial search engine for papers and system documentation will be established early in the project (see D6.1). For an integration into the DKS base we only need to build new harvesters – i.e. programs that generate keywords and formula URL/pairs from the contents (see D6.6). For the data/software components in DKS this is true in principle, but the formulae in code can take many more forms and the notion of a hit URL is not as clear. But the theory graph structure and foundation change morphisms can be integrated into search so that even systems that are incompatible at first glance can be searched under one interface [18].

But this puts high demands on the search interface (user inputs are usually only meaningful with respect to a given context). We will explore this together with the notebook development – semantically annotated notebooks and active documents serve as an explicit context here together with WP4; results of this integration will be reported in D6.10.

Participation per Partner

Partner number and short name	WP6 effort
1 - UPSud	37.00
3 - JacobsUni	46.00
10 - USTAN	10.00

Partner number and short name	WP6 effort
12 - UWarwick	25.00
13 - UZH	12.00
14 - Logilab	2.00
Total	132.00

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D6.1	Full-text Search (Formulae + Keywords) over LaTeX-based Documents (e.g. the arXiv subset)	3 - JacobsUni	Other	Public	9
D6.2	Initial DKS base Design (including base survey and Requirements Workshop Report)	3 - JacobsUni	Report	Public	12
D6.3	Design and implementation of Triform (DKS) Theories	3 - JacobsUni	Other	Public	15
D6.4	Conversion of existing and new Databases (LMFDB, OEIS,FindStat) to unified interoperable System	13 - UZH	Websites, patents filling, etc.	Public	24
D6.5	PYTHON/SAGE Computational Foundation Module in OMDoc/MMT	3 - JacobsUni	Other	Public	24
D6.6	Full-text search (Formulae + Keywords) over CAS Modules and Notebooks	3 - JacobsUni	Other	Public	30
D6.7	PYTHON/SAGE Declarative Semantics in OMDoc/MMT	3 - JacobsUni	Other	Public	36
D6.8	LMFDB Algorithm Verification	3 - JacobsUni	Other	Public	36

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
	with respect to a Triformal Theory				
D6.9	Shared persistent Memoisation Library for PYTHON/SAGE	10 - USTAN	Other	Public	42
D6.10	Search from Notebooks/ Active Documents Interface	3 - JacobsUni	Other	Public	42
D6.11	LMFDB Integration of Algorithms, Data and Presentation	3 - JacobsUni	Report	Public	48

Description of deliverables

D6.1 : Full-text Search (Formulae + Keywords) over LaTeX-based Documents (e.g. the arXiv subset) [9]
Full-text Search (Formulae + Keywords) over LaTeX-based Documents (e.g. the arXiv subset)

D6.2 : Initial DKS base Design (including base survey and Requirements Workshop Report) [12]
Initial DKS base Design (including base survey and Requirements Workshop Report)

D6.3 : Design and implementation of Triform (DKS) Theories [15]
Design of Triform (DKS) Theories (Specification/RNC Schema/Examples) and Implementation of Triform Theories in the MMT API

D6.4 : Conversion of existing and new Databases (LMFDB, OEIS,FindStat) to unified interoperable System [24]
Conversion of existing and new Databases (LMFDB, OEIS,FindStat) to unified interoperable System

D6.5 : PYTHON/SAGE Computational Foundation Module in OMDoc/MMT [24]
PYTHON/SAGE Computational Foundation Module in OMDoc/MMT

D6.6 : Full-text search (Formulae + Keywords) over CAS Modules and Notebooks [30]
Full-text search (Formulae + Keywords) over CAS Modules and Notebooks

D6.7 : PYTHON/SAGE Declarative Semantics in OMDoc/MMT [36]
PYTHON/SAGE Declarative Semantics in OMDoc/MMT

D6.8 : LMFDB Algorithm Verification with respect to a Triformal Theory [36]
LMFDB Algorithm Verification with respect to a Triformal Theory

D6.9 : Shared persistent Memoisation Library for PYTHON/SAGE [42]
Shared persistent Memoisation Library for PYTHON/SAGE

D6.10 : Search from Notebooks/Active Documents Interface [42]
Search from Notebooks/Active Documents Interface

D6.11 : LMFDB Integration of Algorithms, Data and Presentation [48]
LMFDB Integration of Algorithms, Data and Presentation

Schedule of relevant Milestones

Milestone number [18]	Milestone title	Lead beneficiary	Due Date (in months)	Means of verification
MS1	Startup	1 - UPSud	12	By Milestone 1 we will have carried out the requirements study, design and prototype implementations and started community building activities.
MS2	Implementations	1 - UPSud	24	By Milestone 2 we will have constructed first fully functional interface implementations and released enhanced versions of OpenDreamKit components, and train early adopters of Open-DreamKit.
MS3	Community/ Experiments	1 - UPSud	36	By Milestone 3 we will have gathered and evaluated feedback on OpenDreamKit software and established the portfolio of experiments produced with OpenDreamKit through further engaging with the community.
MS4	Evaluation	1 - UPSud	48	By Milestone 4 we will have released final versions of all OpenDreamKit components and completed the project evaluation.

Work package number [9]	WP7	**Lead beneficiary** [10]	6 - UOXF
Work package title	Social Aspects		
Start month	1	**End month**	48

Objectives

The processes by which mathematical knowledge and mathematical software are developed, validated and applied are quite distinctive. In other sciences, the universe provides "ground truth" and the scientific texts or theories can be validated against that by experiment. In mathematics the text itself is the ground truth. The traditional model of mathematical research is a mathematician, or a small group of mathematicians, standing around a blackboard, producing a proof they would "clean up": remove all traces of the process that led to its discovery and then submit the "clean" text to their peers for review.

Mathematicians have adopted new technology in a variety of ways: email and shared documents are used to collaborate on problem-solving and writing; larger "crowdsourcing" [33, 38], arrangements pull together diverse experts; symbolic computation tackles huge routine calculations; and computers check proofs that are too long and complicated for a human to comprehend. These technologies reveal (since email messages, version control systems and bulletin boards can be analysed) and alter the ways in which mathematicians collaborate.

In an EPSRC funded project "The Social Machine of Mathematics" Martin and others are bringing together rigorous methods from the social sciences to study these collaborative processes. Combining this research with the algorithmic game theory expertise of Elkind and Pasechnik, in this work package we intend to pursue the following objectives:

• incorporate the insights from this and similar projects into the design of OpenDreamKit VRE, ensuring that it supports the ways in which mathematicians really work, rather than the way software developers—or indeed mathematicians—think they do;

• extend this work to study the collaborative processes of free open source (mathematical) software development so as to produce guidelines for best practice as well as to develop ideas for extending existing processes to a "system of systems".

Description of work and role of partners

WP7 - Social Aspects [Months: 1-48]
UOXF, USFD, SOUTHAMPTON
"Crowdsourcing"—fine-grained collaborative development of ideas, proofs or software—is a common theme to both objectives. The purpose of a VRE is to allow effective crowdsourcing of computationally supported mathematical results (theorems, proofs, etc.), while free software development is inherently a collaborative process, and we wish to study the best ways of allowing it to scale.

In a sense, mathematics has been a crowdsourced endeavour, dating as far back as the foundation of the Royal Society (UK) in the seventeenth century. The first scientific journals were published collections of letters received, posing questions and observations and offering solutions. Although limited by the speed of physical post, this model had much in common with the public email lists that underpinned collaborative software development in the 1990s. In recent years, the internet and critical tools such as distributed version control have supported much more widespread and finer-grained forms of crowdsourcing, first in software development, and, more recently, in mathematics: examples are provided by online mathematics communities, such as Math-overflow [24] and Polymath Projects [33, 38]. Supporting and encouraging "Mutual crowdsourcing" is the main driving force for developing and maintaining any large-scale open-source virtual research environment.

In this work package we will build on the work of Prof. Martin and her collaborators on the EPSRC project, and, in particular, their study of crowdsourcing, and integrate their findings with tools provided by the burgeoning field of algorithmic mechanism design in order to to optimise crowdsourcing workflows in open-source VREs.

T7.1 Social Science Input to Design M0-M3@.5; M12-M42@.5
Sites: UOXF (lead), UOXF, UPSud

The purpose of this task is to ensure that the design of OpenDreamKit VRE reflects the lessons learned by social scientists studying the ways in which mathematicians actually collaborate and work. Since UO and Martin in particular are already central in the community working in this area, we are well placed to ensure that this happens. As soon as the project begins, team members at UO will combine their own work with a review of the published literature, and identify and meet with key research groups in this area, in order to distill relevant current knowledge for use in the design phases of other parts of the project. They will present the lessons learned at project meetings and workshops and deliver it as a report D7.5 (part I) in month 3.

After that, they will monitor the further development of this area and ensure that any new insights are communicated promptly to the rest of the project. This will be synthesised for archival purposes into two further reports D7.5 in months 24 (part II) and 42 (part III).

We will survey the data needed to assess development models of large-scale academic open-source projects, such as the probable correlation between the size of the atomic contribution vs. the speed of the contribution making it into the code, and collect appropriate statistical data, to be published as a report (and possibly a conference publication) D7.1. The latter will require non-trivial amount of programming work, even only for the test system, SAGE.

T7.2 Implications of VREs for Publication M12-M42
Sites: USFD (lead), UOXF

A key aspect of the OpenDreamKit VRE is support for the full life-cycle of mathematical research, up to, and after publication of results. While it is necessary to support established models of publication, which are central to mathematical practice and academic life, it is also appropriate to explore whether VRE technologies may enable novel models for the distribution of scientific output that are more effective for new forms of mathematical results. The current model for dissemination of scientific output stems from an era when the printing press was dominant. The process has become formalised through peer review and publication of journals. The PDF format widely used for distribution of documents reflects the status quo that a scientific paper is written as if for printing and remains an unchanging document. In scientific blogging we are seeing that more rapid propagation of ideas can occur when the constraints of the printed format are relaxed; however, these dissemination routes lack the formalisation that ensures (usually) fair attribution of ideas and commentary.

We will prototype and evaluate tools and ideas for dissemination of scientific knowledge that do not rely on a static format and allow for the full spectrum of scientific debate. The tools will enable proper credit allocation by encouraging shared attribution of ideas, software and data. This will interact with work in WP6 concerning attribution and citeability for mathematical databases.

Tools to be prototyped include live posters for distribution of knowledge, designed for integration with either large touch screens or smaller tablets D7.4 as well as extensions of the Jupyter project that would provide facilities for commenting on notebooks, which we expect to encourage debate on mathematical and computational ideas D7.3.

T7.3 Mechanism Design for Free Software Development M6-M48@.5
Sites: UOXF (lead)

While crowdsourced open-source software development has become an incredibly powerful force in recent years, it still has limitations. Open source projects tend to be fragile, in the community sense, and suffer from disagreements that ultimately result in “forks” and the resulting duplication of effort. We will analyse this phenomenon in the framework of algorithmic game theory, and try to design finely tuned systems of incentives and rewards for contribution so as to increase the stability of the community and its useful output.

We will focus on three areas:

(1) prioritisation of bug fixes and feature requests to achieve reliable and useful systems;
(2) effective cooperation among multiple collaborative projects;
and (3) making decisions about the strategic direction of the system.

We will use prioritisation as a testbed for designing incentives that encourage all participants to contribute towards sustained development of the most important parts of the system. To this end, we will use ideas from the burgeoning field of mechanism design [26] and in particular recent research on crowdsourcing in algorithmic mechanism design [34]. While doing so, we will apply outcomes to a case study system—SAGE.

The reason why prioritisation poses a challenge in the development of open-source academic software is that this process is task-driven: typically, tasks (also known as tickets) are posted on a website, and their priorities are set in an ad hoc manner. This model is usually good enough for simple bug fixing, but for more elaborate tasks it often leads to unacceptable delays. We will apply preference and opinion aggregation techniques [3] to develop a community prioritisation scheme for bugs and feature requests (which may rely on a reputation scores technique, such as one used on MathOverflow), and implement this scheme as a TRAC [39] add-on D7.2. As SAGE is using the TRAC server [2], this will be easy to test on our testbed system.

Trusting results of computer calculations is crucial for usability; channels for communicating bug reports and fixes need to be carefully analysed from social point of view. Commercial closed-source computer algebra and other computational systems often fail to react to bug reports in a timely manner, and sometimes fall into the short-sighted trap of hiding bugs from potential and current users [7]. Open source systems are only marginally better in this respect, as indicated by recent computer security scares, such as the one around Bash [5]. A game-theoretic analysis of this situation will be attempted.

A key strength of free and open-source software models is the ability to build upon pre-existing software. GAP, PARI/GP, S INGULAR and especially SAGE have made heavy use of this ability. Problems arise over time, however, as priorities of the system developers diverge. For instance, bugs reported by so-called “downstream” systems may not be given the same priority as bugs reported by direct users of the “upstream” system, or ignored altogether; similarly, incompatible changes can be made as long as they are acceptable to the direct user community, even if they cause problems for a dependent system. We will explore how sociological and game-theoretic insights can be used to reduce these problems.

The results of this task will be summarised in D7.7 and reported at relevant AI workshops and conferences.

T7.4 Evaluation of Micromagnetic VRE M32-M44@0.5
Sites: SOUTHAMPTON (lead), UOXF, UPSud

We will use the micromagnetic VRE demonstrator (T4.13), its dissemination workshops (T2.8) and interactions with its users and contributors in the micromagnetic community to evaluate, reflect and report on the project, taking into account technical and social aspects.

A survey will be developed and used to gather user input and feedback on usefulness of the provided capabilities, with particular focus on the capabilities of the micromagnetic VRE to (i) enable new and better science, to (ii) allow to make progress effectively, to (iii) carry out computational science reproducibly, to (iv) collaboratively enable trust and to (v) become a self-sustained project from community contributions. Amongst other channels, we will target attendees of the micromagnetic VRE dissemination workshops (T2.8) to gather data.

All results and insights will be summarised in a public document (D7.8) and reported at appropriate workshops and conferences to share the lessons learned from this JUPYTER-based VRE for micromagnetics. We will create a manuscript for journal publication, summarising the demonstrator project and this evaluation. An important point of this publication is to provide a reference that can be cited by publications making use of the new micromagnetic VRE, to allow tracking of uptake and development of this VRE beyond the life time of this H2020 project.

Participation per Partner

Partner number and short name	WP7 effort
6 - UOXF	23.00
8 - USFD	18.00
9 - SOUTHAMPTON	6.00
Total	47.00

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D7.1	The flow of code and patches in open source projects	6 - UOXF	Report	Public	18
D7.2	TRAC add-on to manage ticket prioritisation	6 - UOXF	Other	Public	24

List of deliverables

Deliverable Number [14]	Deliverable Title	Lead beneficiary	Type [15]	Dissemination level [16]	Due Date (in months) [17]
D7.3	Demo: Mechanism for comments on posted Jupyter notebooks	8 - USFD	Demonstrator	Public	24
D7.4	Demo: Jupyter Notebook Live Poster	8 - USFD	Demonstrator	Public	36
D7.5	Report on relevant research in sociology of mathematics and lessons for design of OpenDreamKit VRE	6 - UOXF	Report	Public	42
D7.6	Review of new publication mechanisms, including evaluation of demonstrator projects	8 - USFD	Report	Public	42
D7.7	Game-theoretic analysis of development practices in open-source VREs	6 - UOXF	Report	Public	42
D7.8	Micromagnetic VRE environment evaluation report	9 - SOUTHAMPTON	Report	Public	48

Description of deliverables

D7.1 : The flow of code and patches in open source projects [18]

The flow of code and patches in open source projects

D7.2 : TRAC add-on to manage ticket prioritisation [24]

TRAC add-on to manage ticket prioritisation

D7.3 : Demo: Mechanism for comments on posted Jupyter notebooks [24]

Demo: Mechanism for comments on posted Jupyter notebooks

D7.4 : Demo: Jupyter Notebook Live Poster [36]

Demo: Jupyter Notebook Live Poster

D7.5 : Report on relevant research in sociology of mathematics and lessons for design of OpenDreamKit VRE [42]

Report on relevant research in sociology of mathematics and lessons for design of OpenDreamKit VRE, parts I-III, with part I (resp. II) due at month 3 (resp. 24)

D7.6 : Review of new publication mechanisms, including evaluation of demonstrator projects [42]

Review of new publication mechanisms, including evaluation of demonstrator projects

D7.7 : Game-theoretic analysis of development practices in open-source VREs [42]

Game-theoretic analysis of development practices in open-source VREs

D7.8 : Micromagnetic VRE environment evaluation report [48]

Micromagnetic VRE environment evaluation report

Schedule of relevant Milestones

Milestone number [18]	Milestone title	Lead beneficiary	Due Date (in months)	Means of verification
MS1	Startup	1 - UPSud	12	By Milestone 1 we will have carried out the requirements study, design and prototype implementations and started community building activities.
MS2	Implementations	1 - UPSud	24	By Milestone 2 we will have constructed first fully functional interface implementations and released enhanced versions of OpenDreamKit components, and train early adopters of Open-DreamKit.
MS3	Community/ Experiments	1 - UPSud	36	By Milestone 3 we will have gathered and evaluated feedback on OpenDreamKit software and established the portfolio of experiments produced with OpenDreamKit through further engaging with the community.
MS4	Evaluation	1 - UPSud	48	By Milestone 4 we will have released final versions of all OpenDreamKit components and completed the project evaluation.

1.3.4. WT4 List of milestones

Milestone number [18]	Milestone title	WP number [9]	Lead beneficiary	Due Date (in months) [17]	Means of verification
MS1	Startup	WP1, WP2, WP3, WP4, WP5, WP6, WP7	1 - UPSud	12	By Milestone 1 we will have carried out the requirements study, design and prototype implementations and started community building activities.
MS2	Implementations	WP1, WP2, WP3, WP4, WP5, WP6, WP7	1 - UPSud	24	By Milestone 2 we will have constructed first fully functional interface implementations and released enhanced versions of OpenDreamKit components, and train early adopters of Open-DreamKit.
MS3	Community/ Experiments	WP1, WP2, WP3, WP4, WP5, WP6, WP7	1 - UPSud	36	By Milestone 3 we will have gathered and evaluated feedback on OpenDreamKit software and established the portfolio of experiments produced with OpenDreamKit through further engaging with the community.
MS4	Evaluation	WP1, WP2, WP3, WP5, WP6, WP7	1 - UPSud	48	By Milestone 4 we will have released final versions of all OpenDreamKit components and completed the project evaluation.

1.3.5. WT5 Critical Implementation risks and mitigation actions

Risk number	Description of risk	WP Number	Proposed risk-mitigation measures
1	Recruitment of highly qualified staff	WP1, WP3, WP4, WP5, WP6	Great care was taken identifying pool of candidates to hire from, and coordinating with currently running projects to rehire personnel with strong track record. Typically, we will rehire European postdocs that are currently funded by the Sloan grant to work on Jupyter in California and wish to come back to Europe.
2	Different groups not forming effective team	WP1, WP3, WP4, WP5	Long track record of working collaboratively on code across multiple sites; Aggressive planning of project meetings, work-shops and one-to-one partner visits to facilitate most effective teamwork, combining face-to-face time at one site with remote collaboration.
3	Implementing infrastructure that does not match the needs of end users.	WP3, WP4, WP6	Most of the members of the consortium are themselves end-users with a diverse range of needs and points of views; hence the design of the proposal and the governance of the project is naturally steered by demand; besides, because we provide a toolkit, users have the flexibility to adapt the infrastructure to their needs.
4	Lack of predictability for tasks that are pursued jointly with the community.	WP3, WP4, WP5, WP6	The PI's have a strong experience managing community-developed projects where the execution of tasks depends on the availability of partners. Some tasks may end up requiring more manpower from Open- DreamKit to be completed on time, while others may be entirely taken care of by the community. Reallocating tasks and redefining work plans is common practice needed to cater for a fast evolving context. Such random factors will be averaged out over the large number of independent tasks.
5	Reliance on external software components	WP3, WP4, WP5, WP6	The non trivial software components OpenDreamKit relies on are open source. Most are very mature and supported by an active community, which offers strong long run guarantees. The critical emerging software component JUPYTER builds on IPYTHON which has been around for a decade and is very mature. The other components could be replaced by alternatives, or worst comes to worst, taken over by the participants.

1.3.6. WT6 Summary of project effort in person-months

	WP1	**WP2**	**WP3**	**WP4**	**WP5**	**WP6**	**WP7**	**Total Person/Months per Participant**
1 - UPSud	26.50	9	50	12	6	37	0	140.50
2 - CNRS	2	19	0	28	40	0	0	89
· UB	0	0	0	0	0	0	0	0
3 - JacobsUni	2	0	0	12	0	46	0	60
4 - UJF	2	0	6	0	52	0	0	60
5 - UNIKL	2	2	0	2	60	0	0	66
6 - UOXF	2	0	4	0	0	0	23	29
7 - USlaski	2	30	0	4	0	0	0	36
8 - USFD	2	17	0	6	12	0	18	55
9 - SOUTHAMPTON	2	16	6	16	0	0	6	46
10 - USTAN	2	18	16	18	18	10	0	82
11 - UVSQ	2	2	8	2	0	0	0	14
12 - UWarwick	2	0	0	0	0	25	0	27
13 - UZH	1	0	0	0	0	12	0	13
14 - Logilab	2	6	14	12	12	2	0	48
15 - Simula	2	2	0	28	0	0	0	32
16 - UGent	1.50	1	14	14	0	0	0	30.50
Total Person/Months	55	122	118	154	200	132	47	828

1.3.7. WT7 Tentative schedule of project reviews

Review number [19]	**Tentative timing**	**Planned venue of review**	**Comments, if any**
RV1	6	Brussels	interim review
RV2	18	Brussels	
RV3	36	Brussels	
RV4	48	Brussels	

1. Project number

The project number has been assigned by the Commission as the unique identifier for your project. It cannot be changed. The project number **should appear on each page of the grant agreement preparation documents (part A and part B)** to prevent errors during its handling.

2. Project acronym

Use the project acronym as given in the submitted proposal. It can generally not be changed. The same acronym **should appear on each page of the grant agreement preparation documents (part A and part B)** to prevent errors during its handling.

3. Project title

Use the title (preferably no longer than 200 characters) as indicated in the submitted proposal. Minor corrections are possible if agreed during the preparation of the grant agreement.

4. Starting date

Unless a specific (fixed) starting date is duly justified and agreed upon during the preparation of the Grant Agreement, the project will start on the first day of the month following the entry into force of the Grant Agreement (NB : entry into force = signature by the Commission). Please note that if a fixed starting date is used, you will be required to provide a written justification.

5. Duration

Insert the duration of the project in full months.

6. Call (part) identifier

The Call (part) identifier is the reference number given in the call or part of the call you were addressing, as indicated in the publication of the call in the Official Journal of the European Union. You have to use the identifier given by the Commission in the letter inviting to prepare the grant agreement.

7. Abstract

8. Project Entry Month

The month at which the participant joined the consortium, month 1 marking the start date of the project, and all other start dates being relative to this start date.

9. Work Package number

Work package number: WP1, WP2, WP3, ..., WPn

10. Lead beneficiary

This must be one of the beneficiaries in the grant (not a third party) - Number of the beneficiary leading the work in this work package

11. Person-months per work package

The total number of person-months allocated to each work package.

12. Start month

Relative start date for the work in the specific work packages, month 1 marking the start date of the project, and all other start dates being relative to this start date.

13. End month

Relative end date, month 1 marking the start date of the project, and all end dates being relative to this start date.

14. Deliverable number

Deliverable numbers: D1 - Dn

15. Type

Please indicate the type of the deliverable using one of the following codes:

R	Document, report
DEM	Demonstrator, pilot, prototype
DEC	Websites, patent fillings, videos, etc.
OTHER	
ETHICS	Ethics requirement

16. Dissemination level

Please indicate the dissemination level using one of the following codes:

PU	Public
CO	Confidential, only for members of the consortium (including the Commission Services)
EU-RES	Classified Information: RESTREINT UE (Commission Decision 2005/444/EC)
EU-CON	Classified Information: CONFIDENTIEL UE (Commission Decision 2005/444/EC)
EU-SEC	Classified Information: SECRET UE (Commission Decision 2005/444/EC)

17. Delivery date for Deliverable

Month in which the deliverables will be available, month 1 marking the start date of the project, and all delivery dates being relative to this start date.

18. Milestone number

Milestone number:MS1, MS2, ..., MSn

19. Review number

Review number: RV1, RV2, ..., RVn

20. Installation Number

Number progressively the installations of a same infrastructure. An installation is a part of an infrastructure that could be used independently from the rest.

21. Installation country

Code of the country where the installation is located or IO if the access provider (the beneficiary or linked third party) is an international organization, an ERIC or a similar legal entity.

22. Type of access

VA	if virtual access,
TA-uc	if trans-national access with access costs declared on the basis of unit cost,
TA-ac	if trans-national access with access costs declared as actual costs, and
TA-cb	if trans-national access with access costs declared as a combination of actual costs and costs on the basis of unit cost.

23. Access costs

Cost of the access provided under the project. For virtual access fill only the second column. For trans-national access fill one of the two columns or both according to the way access costs are declared. Trans-national access costs on the basis of unit cost will result from the unit cost by the quantity of access to be provided.