CodeRefinery project plan (v1.0)

2016–10–28

Contents

Table 1: Version history

Version	Date	Comments
v0.0	2016–05–01	First draft
v0.1	2016–05–19	Second draft
v0.2	2016–06–13	Third draft
v0.3	2016–07–13	Fourth draft
v0.4	2016–08–22	Fifth draft
v0.5	2016–10–21	Sixth draft
v1.0	2016–10–28	First approved version

Approval of the project plan

This project plan version v1.0 has been approved by the steering group on 2016–10–28.

1 Idea and objective

1.1 Background

Software is transforming research in a fundamental and disruptive way and the quality of research software is increasingly becoming critical for the advancement of knowledge, in particular for interdisciplinary research.

However, developers of research software face a number of challenges for which many researchers have not been trained: They need to keep their code a) flexible, extensible, performing, and scaling with respect to system size in order to be able to explore new scientific territories, b) usable and adaptable to new algorithms, new hardware, new software, and new libraries in order for the code to remain portable, and finally c) keep their code as simple and understandable as possible to prevent their codes from becoming unmaintainable.

It is particularly the last point, code complexity, which can decide the success of a research code project and can have a decisive impact on the scientific community. Code complexity and suboptimal tools collaboration routines can make projects stall or even fail.

Experience from teaching and working with scientific code projects shows that scientists often lack the necessary training in practices and infrastructure to enable them to collaboratively write high-quality scientific software. This means that many academic research groups develop software using inefficient or outdated methodologies which often do not scale to the size of the research group or project collaboration and which often lead to irreproducible results and unmaintainable code. In such programs new features become harder and harder to implement and the development speed decreases over time or even halts.

One reason for this is that many packages have started as one-person pet projects aimed at solving a specific problem at hand. But over the years the projects would outgrow the code layout and programming style with many new developers joining the team and the codebase expanding considerably. Very often the core developers have never received training in modern software development methodologies because their main training and interest is in their respective scientific domain. It is often observed that decades-old and now outdated programming techniques are employed and passed from senior to junior developers joining the team without questioning or revising the tools and techniques.

Many research groups struggle with tens or hundreds of thousands of lines of complex legacy code which make it almost impossible to move the code forward in terms of new functionality or efficiency.

From related training events, discussions, and collaboration with several scientific programs the need for relevant training in modern software development methodologies and the need for access to a modern software development e-infrastructure including necessary technical expertise is recognized to be huge. This project addresses this need.

1.2 Project goal

The goal of this Nordic e-Infrastructure Collaboration [1] project is to provide researchers with infrastructure and training in the necessary tools and techniques to create sustainable, modular, reusable, and reproducible software.

The result of this project is a set of software development e-infrastructure solutions, coupled with necessary technical expertise and extensive training and on-boarding activities, training material and best practices guides which together form a Nordic platform for research groups and institutes to develop a better collaboration on software and thereby to catalyze reproducible research and collaboration.

1.3 Project scope

The main aspect of this project is to train Nordic research groups and projects in state-of-the-art tools and practices in modern software design and development. In order to quickly bring researchers up-to-speed, the project will organize a series of training events across the main nodes in the Nordic countries. The events will be organized as close to users as possible, e.g. at research institutions with large communities where the biggest impact can be achieved. An important ingredient of the training aspect will be the creation of best practices guides which will be open to the entire user community and reviewed by the community to ensure quality and relevance. The project members will train groups in building modular and maintainable code and encourage open science and best practices towards reproducible science. This project will assist groups in migrating from ad-hoc in-house infrastructure and solutions to collaborative industry-strength community solutions encouraging code review and automated testing.

The second aspect of this project is to provide the necessary systems and tools and deploy necessary infrastructure where otherwise available community solutions do not exist. These will include:

- Distributed version control repository hosting for non-public code to allow for a very efficient collaborative code development work-flow.
- Issue tracking system which is a necessity for any software project (ideally integrated with a web-based repository hosting).
- Code review system which will improve the quality of the code as well as the skills of programmers and communication among team members (ideally integrated with a web-based repository hosting).
- Continuous integration platform to automatically build software and test code changes in order to detect code regression as early as possible (ideally coupled with the code review system).
- Platform for the distribution of free of charge binary code where sources are not public or distribution of free of charge sources under a custom license to increase the visibility and impact of Nordic software projects and reduce the distribution and maintenance burden both for developers and users (optional work package).

Finally, the third aspect of the CodeRefinery project will be to build a community of users and experts by aggregating and providing courses and organizing training, summer schools, meet-ups, tutorials and forums and engaging in relevant social media. By building on the experience of a Nordic-wide network of application experts, the project can create the nucleus of a wider communication environment for exchange of expertise.

The project includes 10 project members (summing up to 2.25 FTE over the duration of two years) as well as a project leader (0.5 FTE over the duration of two years).

1.4 Objective priority

Top priority is to educate the researcher in how to best use available tools in modern software development. While actual tools will probably change in future and best practice guides may become outdated, the learned best practices can be adapted to new tools and work-flows.

The duration of the project is planned to be 2 years which forces us to quickly produce results and to get communities on board early in the project in order to generate a critical mass of users and to maintain the motivation to fund the project.

In terms of numerical values the priority (summing to 1) is defined in Table 2.

Table 2: Objective priority

Objective	Priority
Result	0.6
Time	0.3
Cost	0.1

1.5 Success factors

The most important success factors are the following:

- Availability of personnel with relevant competence for project work.
- Close interaction with user communities through training and pilot projects to make sure that the result is needed and relevant.
- Visibility in the user communities and engaging the user communities early in the project.
- Choice of infrastructure solutions which can continue to run and remain maintained beyond the duration of the project.
- Availability of hardware resources and relevant cloud services from national e-infrastructure providers.
- Open, transparent, and effective communication.
- Effective project management.

1.6 Expected benefit

Adoption of modern tools and best practices for software development will bring considerable benefits to researchers by increasing:

- productivity of the researchers
- quality of their codes (in terms of flexibility, viscosity, entropy)
- scientific reproducibility
- collaboration possibilities
- number of derivative products (through easier collaboration)
- skills of the developers
- attractiveness of software projects to talented programmers
- visibility of programs through simpler distribution

National providers will benefit from 1) the increased competence among their users and developers, which will lead to 2) improved utilization of their resources through development of better software, as well as 3) the long-term maintainability of locally developed software.

The Nordic region as a whole will benefit from the nucleation of a community and increased interactions between application experts from different countries, as well as with the corresponding user groups.

1.7 Limitations

This project is about tools and practices, not about implementation of algorithms. This project will not replace the programming effort which stays with the researchers. The scope of this project is to assist researchers with relevant tools and with the migration effort towards the relevant tools. It is also not the scope of the project to optimize code for performance or to compete with advanced user support activities but it may be in the scope of this project to teach relevant tools and practices to be able to analyze and optimize code performance. In other words this project is not about assisting user communities to write more efficient code but rather about assisting in writing code more efficiently. This project is mostly about optimizing human time, not CPU time.

This project is also not a replacement or in competition to the Software Carpentry initiative but rather a complementary element in education. This project will not consider teaching basic programming in specific languages but focus on tools and routines to leverage programming skills in a collaborative setting. In this project we focus on code complexity and sustainability which is complementary to the typical curriculum of the Software Carpentry initiative.

1.8 Interdependencies to other organizations or projects

This project will network, leverage, collaborate with, and contribute to a number of related projects in order to increase visibility, impact, and relevance. Below, the related projects are briefly highlighted.

NeIC Pooling Competencies Focus Area: The goal of this NeIC project is to map skills, identify, and prioritize the needs of user communities and to establish cross-border training programs across the Nordic countries.

Software Carpentry: Since 1998, Software Carpentry has been teaching researchers in science, engineering, medicine, and related disciplines the computing skills they need to get more done in less time and with less pain. Its volunteer instructors have run hundreds of events for thousands of scientists in the past two and a half years (see http://software-carpentry.org).

The Software Sustainability Institute: A UK-national facility for cultivating and improving research software to support world-class research (see http://www.software.ac.uk).

The Molecular Sciences Software Institute: The Molecular Sciences Software Institute will promote improved interoperability of community codes, easier deployment on heterogenous computing architectures, greater parallel scalability of existing and emerging theoretical models, as well as substantially improved training of molecular-science students in software design and engineering. Through a range of multi-institutional outreach efforts the Institute will engage the community to increase the diversity of its workforce by more effectively attracting and retaining students and faculty from underrepresented groups (see http://molssi.org).

Working towards Sustainable Software for Science Practices and Experiences (WSSSPE): WSSSPE is an international community-driven organization that promotes sustainable research software by addressing challenges related to the full lifecyle of research software through shared learning and community action (see http://wssspe.researchcomputing.org.uk).

2 Tasks

2.1 Task 0: Management and coordination

Resource need: 4 person-months (PMs).
Content: Communication, coordination, scheduling, and participation in meetings, and preparation of documents and proposals related to the CodeRefinery project as well as training the trainers.

2.2 Task 1: Project website and outreach

Resource need: Initial setup: 0.5 PMs, maintenance: 3 PMs.
Content: Create a project website which will be the main entry point to learn about the CodeRefinery project and to access the training material, best practices guides, and relevant services. To allow for quality review by the user community, the sources of the website will be open and derivative work and external contributions will be technically possible and explicitly encouraged. To avoid duplicating content between the project website and the NeIC website, the latter will generally feature more static content, present the project team and serve project documents relevant in the NeIC context.
Components:

- 1.1 - Reserve domain (http://coderefinery.org)
- 1.2 - Set up a virtual machine to host web services
- 1.3 - Create a website which contains general presentation of the project goals, partners, and project team members, and put it in production
- 1.4 - Present the project
- 1.5 - Present the project members (both on http://coderefinery.org and the standard NeIC website)
- 1.6 - Add links to related projects
- 1.7 - Add blog functionality for best practices guides
- 1.8 - Integrate events calendar with Pooling Competencies NeIC training calendar
- 1.9 - Reserve and maintain a Twitter account

Delivery date: Pilot version of the website deployed by September 2016 and edited and improved continuously after.

2.3 Task 2: Training material

Resource need: 12 PMs.
Content: Development of training material in terms of web material, guides, and exercises which will be used in CodeRefinery training courses but can also be employed in self study. The material will be hosted on GitHub [2] and deployed to GitHub Pages [3] and Read the Docs [1]. A starting point will be to migrate and update existing training material from the SeSE course "Scientific software development toolbox" (licensed under CC BY 4.0) and then continuously expand and improve it. The examples used in the training material will be based on freely available services where possible and applicable. For the cases where such services are not available or applicable, the training material will use the source code repository hosting (Task 5) as examples. In the situation where relevant training material is already available, we link to it instead of duplicating information.
Components:

- 2.1 - Managing complexity in software development
- 2.2 - Introduction to version control and Git
- 2.3 - Branching, merging, and conflict resolution in Git

[1] Read the Docs is a service which serves open source so-called restructured text or Markdown-based documentation free of charge, see http://readthedocs.org.

- 2.4 - Collaboration via distributed version control and code review
- 2.5 - Working with GitHub
- 2.6 - Working with GitLab
- 2.7 - Git branch design and release strategies
- 2.8 - Modern code documentation
- 2.9 - Automated testing
- 2.10 - Build tools
- 2.11 - Mixed language development
- 2.12 - Modular development and library design
- 2.13 - Building multi-component software
- 2.14 - Profiling and code optimization (aggregating references to available material)
- 2.15 - Debugging tools
- 2.16 - Software licensing and packaging
- 2.17 - DevOps and automation

Delivery date: Pilot versions published on the project website December 2016 and edited and improved continuously after. All training material online by December 2017.

2.4 Task 3: Best practices guides

Resource need: 10 PMs.
Content: Creation, publishing, and iterative review of best practices guides targeting self study. In the situation where relevant best practices guides are already available, we link to them instead of duplicating information.
Components:

- 3.1 - Best programming practices in Fortran
- 3.2 - Best programming practices in C and C++
- 3.3 - Best programming practices in Python
- 3.4 - Best practices in Git
- 3.5 - Collaboration via distributed version control and code review
- 3.6 - Git branch design and release strategies
- 3.7 - Modern code documentation
- 3.8 - Automated testing
- 3.9 - Deploying continuous integration
- 3.10 - Migrating a project from hand-written Makefiles to portable build tools

- 3.11 - Best practices in mixed language development
- 3.12 - Modular development and library design
- 3.13 - Building multi-component software
- 3.14 - Profiling and code optimization best practices
- 3.15 - Best practices in debugging
- 3.16 - Best practices for packaging and distribution
- 3.17 - How to open source a project and maintain it afterwards
- 3.18 - Efficient use of cloud services with DevOps
- 3.19 - Setting up a software management plan

Delivery date: Pilot versions published on the project website December 2016 and edited and improved continuously after. All guides online by December 2017.

2.5 Task 4: Training events

Resource need: 12 PMs.
Content: Organization and delivery of training events. These will be both standalone and back-to-back events partnering within the NeIC Pooling Competencies Focus Area and Software Carpentry events. Apart from generic training events the CodeRefinery project will also offer on-demand training for specific user groups as well as one-on-one training and consulting based on an open call and review process. Each training event will be 2 to 3 days, given by 3 teachers.
Training schedule (12 events):

- Dec 2016 - Helsinki
- Feb 2017 - Stockholm
- May 2017 - Copenhagen
- May 2017 - Umeå
- Jun 2017 - Tromsø
- Oct 2017 - Aarhus
- Nov 2017 - Linköping
- Dec 2017 - Helsinki
- Feb 2018 - Trondheim
- Mar 2018 - Turku
- Apr 2018 - Odense
- May 2018 - Uppsala
- Jun 2018 - Oslo

Feedback in the form of an online survey will be an important part of the training events and will help to create a map and list of contacts, groups, projects, and institutions which develop research software in the Nordics and their current development techniques and needs.
Delivery date: First training events start in 2016 and continue throughout the project.

2.6 Task 5: Source code repository hosting

Resource need: Initial setup: 1.5 PMs, maintenance: 2.7 PMs.
Content: To deploy and offer a state of the art web-based source code repository platform for private repositories. The goal of this task will be to offer unlimited private repositories for scientific projects centered in the Nordic countries and to enable and strengthen international collaborations by enabling external collaborators to access the system. Essential components will be issue tracking and integrated code review possibility. Code review is a mechanism where code changes are reviewed and discussed online on the web front-end to the repository before they are integrated into the main development line. It is possible (but not necessary) to set up fine-grained permission levels to prevent developers from bypassing code review and give only experienced developers the permission to integrate changes from contributors.

In order to choose the specific service platform we will form a review board in order to compare alternative and choose the most appropriate for this project and the Nordic community.
Milestones:

- 5.1 - Review technical and administrative alternatives
- 5.2 - Formalize authentication process
- 5.3 - Launch for early adopting projects
- 5.4 - Official launch

Delivery date: Review and design December 2016 to February 2017, deployment March 2017, opening May 2017 for early adopting projects. Official launch in June 2017.

2.7 Task 6: Continuous integration (CI) service

Resource need: Initial setup: 3 PMs, maintenance: 6 PMs.
Content: The project will offer a continuous build and test service which can be used in combination with the source code repository hosting platform (Task 5). This will significantly simplify developing features in parallel in a collaborative setting where each code change can be tested before it is integrated to the main development line without requiring the code maintainers to manually verify the submitted code changes. Part of this task will be to research concrete options for CI deployment. Nordic research groups will be able to participate in this service through an open call and review process. We will document our choice of services and tools and our workflow such that research groups with own hardware will be able to deploy CI using our recommended workflow. This task will nicely integrate with the course material and the best practices guide where we will apply and document DevOps and automation techniques and the efficient use of cloud services with DevOps.
Milestones:

- 6.1 - Migrate http://testboard.org[2] under CodeRefinery
- 6.2 - Research concrete options for CI deployment
- 6.3 - Formalize authentication process
- 6.4 - Deploy CI service
- 6.5 - Invite open call for proposals to join the service

[2]CDash nightly testing dashboard currently maintained by Radovan Bast and hosting test results for a number of quantum chemistry codes.

- 6.6 - Make CI service available to few use cases selected by open call
- 6.7 - Adapt service after collected experiences
- 6.8 - Document the chosen services, tools, and workflow

Delivery date: Review and design March to August 2017, opening September 2017.

2.8 Task 7: Refined delivery plan

Resource need: 0.5 PMs.
Content: The detailed project plan and priorities are set iteratively and need to be revisited after every few months based on previous experiences and use cases. Therefore, the project delivery plan needs to be revisited and updated in the mid-term of the project.
Delivery date: August 2017.

2.9 Task 8: Final report

Resource need: 0.5 PMs.
Content: End of project report is to be prepared before the end of the project followed by a hand-over of services to national e-Infrastructure providers.
Delivery date: August 2018.

2.10 Optional task A: Source and binary distribution platform

Resource need: 4.2 PMs.
Content: The go-to distribution platforms for open-source code are typically GitHub, GitLab, or BitBucket. However, for free of charge closed-source code binary distributions or for free of charge custom-license source distributions such a general platform is missing and SourceForge is typically not a good alternative because it does not provide the program authors with useful metrics other than number of downloads, because it allows anonymous downloads and does not allow program authors to review download requests with a custom license agreement. Another downside of SourceForge is that it is littered with commercials which leaves a bad impression on users seeking to download research software. The consequence of this is that program authors often either program and host their own "home-cooked" distribution websites where registered users can download the offered programs or they distribute code via email or DropBox or they do not distribute their code at all.

The goal of this work package is to create a clean and simple platform which will make it easy for scientists to distribute their free of charge code to the community by lifting the burden of creating and maintaining websites for user registration and downloads under custom license agreements. Program authors will be able to upload their code together with custom license agreements. Software users will be able to register at the website and either download a specific code or request to download a specific code. The code owners will then be able to validate downloads and optionally request information from the software users. Old versions will not simply disappear which will improve the reproducibility of published scientific results.

This will not only simplify maintenance but also offer more reliable metrics for scientists and funding agencies and increase visibility for the scientific codes. The download metrics, usage statistics, and increased visibility of projects will generate a measurable value for the Nordic software community.

To increase visibility even further, this task will consider the possibility to assign a digital object identifier (DOI) to code distributions by partnering with the Zenodo service (http://zenodo.org) or similar services to achieve citeable programs.

Milestones:

- A.1 - Define requirement specifications
- A.2 - Explore technical alternatives to store the data and host the website application
- A.3 - Implementation of a prototype web app
- A.4 - Open the prototype to a selected group of friendly, enthusiastic, and forgiving beta testers
- A.5 - Website launches officially and is available to the user community

Delivery date: Design March and April 2017, prototype May 2017, official launch November 2017.

3 Schedule and resource need

Table 3: Distribution of person-months over the project life span

Month	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	Sum
Task 0	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2			0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2			4
Task 1	0.5	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15		0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15			3.5
Task 2	1	1	1	1	1	1	0.5	0.5	0.5	0.5			0.5	0.5	0.5	0.5	0.5	0.5	0.25	0.25	0.25	0.25			12
Task 3			1	1	1	1	1	1	0.5	0.5			0.5	0.5	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25			10
Task 4				1		1			1	1				1	1	1		1	1	1	1	1			12
Task 5				0.5	0.5	0.5	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	4.2
Task 6							0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	9
Task 7											0.25	0.25													0.5
Task 8																							0.25	0.25	0.5
Task A							0.5	0.5	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	4.2
Sum	1.7	1.35	2.35	3.85	2.85	3.85	3	3	3.2	3.2	1.1	1.25	2.2	3.2	2.95	2.95	1.95	2.95	2.7	2.7	2.7	2.7	1.1	1.1	59.9

Task 0: Management and coordination
Task 1: Project website and outreach
Task 2: Training material
Task 3: Best practices guides
Task 4: Training events
Task 5: Source code repository hosting
Task 6: Continuous integration (CI) service
Task 7: Refined delivery plan
Task 8: Final report
Task A: Source and binary distribution platform

4 Project organization

The project is organized according the PPS project method [4] with the following groups or roles:

- **Steering group**: The steering group takes the major decisions in the project. The steering group accepts the project plan, approves deliveries, and decides on transferal of responsibility of the project results. The steering group also has the possibility to modify the project plan.
- **Project owner**: The project owner is responsible for the practical level project follow-up. The project owner is the chair of the steering group. The Project Owner will oversee that the project works towards the project idea and project objective.
- **Project leader**: The project leader prepares the project plan and is responsible for the day to day management of the project and of leading the project group. The project leader makes proposals for the steering group on decision points, such as acceptance of deliveries, change requests or any running business that needs to be decided. The project leader acts as a secretary in the steering group meetings.
- **Project group**: The project group is formed by the experts contracted for work in the project. The project group is distributed in four countries and several locations. Each project group member assumes responsibility for a certain task in order to level work load and to increase identity with the project.

5 Working methods

5.1 Requirement dialogue

The requirements for the project are acquired from the project directive which has been submitted to and approved by the NeIC board. The project directives are supplemented and adjusted during the course of the project by the steering group. The experts working in the project will refine the requirements to a more technical level. Finally, the use cases to be implemented are a direct route to learn user requirements as they document a certain workflow that the end user wants to accomplish. The requirements will be regularly discussed at the project meetings and steering group meetings.

5.2 Change management

Change requests which require a significant change in the project requirements and the project plan are documented in writing. The decision on approval of change requests will be made by the project leader with consultation to project owner or in a case of major change will be taken to the steering group for decision.

Small changes can be accommodated within the normal development cycle of the project. The project leader will follow the development of the project and if a significant factor affecting the project is emerging, may decide to re-prioritize the requirements accordingly.

5.3 Internal information distribution and document management

- Day to day discussion takes place on the #coderefinery channel on NeIC Slack: https://neic.slack.com.
- Weekly virtual coordination meetings between the project owner and the project leader: minutes stored in NeIC internal wiki pages.

- Initially weekly, later Bi-weekly virtual coordination meetings within the project team: minutes are stored and visible to all team members.
- Kick-off meeting at the beginning of the project.
- Project review meetings every 3-4 months.
- Steering group meetings every 2-4 months.

5.4 External information distribution and outreach

The project will strive for open and transparent communication. The main hub for communications is the project website.

- Project directive and project plan are publicly available online.
- Steering group meeting minutes are publicly available online after being approved by all meeting attendees.
- Public files available through http://coderefinery.org.
- News and events are announced via Twitter.

5.5 Administrative routines

- Service contracts will be made for persons working in the CodeRefinery project. The contract will be made between the employing organization and NeIC. The service contract specifies the amount of working time committed for the project. Formally, the project leader decides on nominating persons to work for the project.
- The project personnel should follow the practices of their home organization for time reporting. The CodeRefinery project does normally not require time reports. Instead, the results will be closely followed. However, time tracking is encouraged for personal use to improve focus and simplify time reporting within the home organizations.
- Team members track the progress of work items and issues in a Trello board which is accessible to all project team members as well as the project owner.
- NeIC will cover costs for travel which is necessary for the project work. Covering the travel costs need to be agreed beforehand with the project leader. The project leader manages the travel budget following the regulations of the home organisations.
- Project leader reports expenses to the project owner upon request.

5.6 Environment and economy

The project attempts to minimize environmental impact and costs by having online meetings instead of physical meetings whenever possible to minimize travel and costs.

6 Risks

Table 4: List of risks

Id	Description	Probability	Impact	Responses	Responsibility
1	Key persons or required competences not available.	medium	high	Identify gaps early and start finding replacement and/or train project members.	Steering group and project leader
2	Insufficient interest from user communities.	low	medium	Involve user communities, travel, announce, spread the word, engage social networks.	Project leader
3	Project life time too short for projects to engage.	high	high	Choose open solutions over lock-in. Make it easy to migrate away. Document migrations.	Project leader
4	Migration barrier for projects is too high.	medium	medium	Document migrations, train, lower the migration barrier as much as possible.	Project leader
5	Insufficient engagement from the e-infrastructure providers in terms of service maintenance.	medium	high	Load-balance the maintenance effort among the four partners. Generate and demonstrate value in the services provided.	Steering group and project owner
6	Authentication solution excludes collaborators outside the country or outside the Nordics.	medium	high	Pragmatic solution which strikes a good balance between security and accessibility.	Project leader

Table 5: Annual salary costs

Project experts	2.00 FTE personnel costs
SGAS[a] maintenance	0.25 FTE personnel costs
Project leader	0.50 FTE personnel costs

[a]see Section 7.1.

7 Project cost estimate

The main cost item of the project is the personnel cost. The volume of the project will be 2.75 FTE annually for two years (Table 5). Other costs for the project are travel costs, meeting costs, costs for domain name and certificate registration, trademark registration, external source code repository hosting, and promotional material (t-shirts and stickers), see Table 6.

Table 6: Budget for costs covered by NeIC (except salary costs; all values in NOK)

	2016	2017	2018	Sum
Training events[a]	23400	163800	117000	304200
Steering group meetings[b]	56000	56000	28000	140000
Kick-off meeting[c]	80840			80840
Project group meetings[d]		124800	62400	187200
Conference budget	30000	60000	60000	150000
Domain name registration	600	600	600	1800
SSL certificate registration	2000	2000	2000	6000
Trademark registration	2000			2000
Repository hosting		20000	14000	34000
CI runner		20000	14000	34000
Stickers (promotion)	2000	2000	2000	6000
Sum	196840	449200	300000	946040

[a]1 training event: travel for 3 persons (3 × 3000 NOK) and accommodation for 3 persons and 4 nights with 1200 NOK/night.
[b]Board and conference room 1000 NOK per person; travel per person: 3000 NOK; number of attendees: 7.
[c]Lunch to next late afternoon; conference room: 3700 NOK/day; accommodation and food per person: 3120 NOK; travel per person: 3000 NOK; number of attendees: 12.
[d]Lunch to next late afternoon; accommodation and food per person: 2200 NOK; travel per person: 3000 NOK; number of attendees: 12.

7.1 SGAS maintenance scope

SGAS is an accounting system for computing systems which is available under an open-source license and currently used to record resource usage for the Nordic participation in the WLCG collaboration (the distributed Nordic Tier-1 and the national Tier-2s) and in some national infrastructures in the Nordics.

7.1.1 The Scope of the service

The parties agree to the following:

- Maintenance of SGAS (including bug report follow-ups, release management, advanced technical support in general and towards related development projects in particular, representation in standardisation activities).

- Acting as a use case within the CodeRefinery project (including participation in surveys, giving feedback to suggested tools, being active in common discussion fora and possibly acting as role model for other software projects).

7.1.2 Liabilities

The responsibility of the CodeRefinery project with regards to SGAS concerns the migration of the SGAS code-base to the CodeRefinery infrastructure and successful application of the CodeRefinery services and tools to further the development and maintenance of the SGAS software. The CodeRefinery project assumes no responsibility for operating SGAS services and infrastructure.

8 References

References

[1] `https://neic.nordforsk.org`

[2] `https://github.com`

[3] `https://pages.github.com`

[4] `https://www.tieto.com/services/consulting-services/change-and-transformation-services/pps-practical-project-steering`