Hardening OpenRefine's Extension Architecture

This application focuses on the goal post Better Support for OpenRefine Extensions. Unlike the Plugin Manager application, this focuses on understanding the tradeoffs of the various options discussed in the Improving the UX of extension install, and Butterfly thread.

Whereas the Plugin Manager UI is largely user-centric, the discovery work discussed in this application is geared towards an improved developer experience for extension developers and OpenRefine maintainers.

Contact information

Your name	Martin Magdinier
Email address	martin@openrefine.org
Phone number	+1-503-383-1430
Organisation	Code For Science and Society
Country	USA

General project information

Proposal name	Hardening OpenRefine's Extension Architecture
Website / wiki	https://openrefine.org

Please be short and to the point in your answers; focus primarily on the what and how, not so much on the why. Add longer descriptions as attachments (see below). If English isn't your first language, don't worry — our reviewers don't care about spelling errors, only about great ideas. We apologise for the inconvenience of having to submit in English. On the up side, you can be as technical as you need to be (but you don't have to). Do stay concrete. Use plain text in your reply only, if you need any HTML to make your point please include this as attachment.

Abstract: Can you explain the whole project and its expected outcome(s). (1200 characters)

OpenRefine provides a foundation for developers to extend the software’s core functionality. There are many points for extension, with more being added in recent years. Meanwhile, OpenRefine itself continues to evolve, and new extensions are continually being developed. The combined growth of OpenRefine and the extension ecosystem has put additional strain on maintainers. There is a consensus within the developer community on some key points of action that can remediate some of this burden.

This project will provide additional tooling to support both extension developers and core OpenRefine maintainers. This would be done by providing documentation of extension points, automated checks for API breakages, and extension isolation. Extension isolation in particular would improve the resilience of OpenRefine, as it would decrease the likelihood of one extension causing problems for the rest of the system.

This project will also unblock future improvements. Upgrading key dependencies is currently infeasible due to the heavy maintenance burden it would place on extension maintainers. The OpenRefine ecosystem will be more sustainable once this burden is removed.

Have you been involved with projects or organisations relevant to this project before? And if so, can you tell us a bit about your contributions? (optional, help to determine we are the right person to take on this project 2500 characters)

This project is led by the OpenRefine core team and has received widespread community support.

OpenRefine (lead applicant) is fiscally sponsored by Code for Science and Society, a 501(c)(3) charitable organization in the USA. OpenRefine leads the development and sustainability of the tool, including maintenance planning, technical debt reduction, contributor onboarding, documentation improvements, maintainability, and community support. The project is led by Rory Sawyer, Developer and Contributor Engagement Lead, and Martin Magdinier, Project Manager.

We have secured partnerships with the following European institutes:

North Rhine-Westphalian Library Service Centre (hbz) relies heavily on OpenRefine as a stable and versatile third-party tool for our customers and partners to comfortably reconcile with and reuse the data they provide. In cooperation with the German National Library (DNB) HBZ provides the reconciliation service for the Integrated Authority File (GND) at https://reconcile.gnd.network which is heavily used by memory institutions and researchers in the DACH region, almost exclusively through OpenRefine. (See e.g. the results from a poll in 2022: https://blog.lobid.org/2022/07/19/survey-results.html.) HBZ is currently developing an OpenRefine extension to update or create GND records after reconciliation. HBZ will directly benefit from an improved OpenRefine extension architecture.

NFDI4Culture, the consortium for research data on material and immaterial cultural heritage within the German National Research Data Infrastructure, relies heavily on OpenRefine to establish a needs-based infrastructure that serves our community of interest, ranging from architecture, art history and musicology to theatre, dance, film and media studies. They have allocated staff time to pilot testing, user feedback sessions, and documentation review. They will also or co-host at least one webinar in Germany to disseminate project results.

We also received a letter of support from **LaOficina Producciones Culturales** regarding their usage of OpenRefine.

Requested support

Explain what the requested budget will be used for?

Does the project have other funding sources, both past and present?

(If you want, you can in addition attach a budget at the bottom of the form)

Explain hardware, labor (including rate), travel cost , technical meeting - max 2500 characters

Budget:

Team Organization

- **Martin Magdinier:** Project Management: hourly rate EUR 50
- **Rory Sawyer:** Development: hourly rate EUR 65
- **Code for Science and Society** fiscal sponsorship fees (15% of total grant)

Hardware & Infrastructure: No dedicated hardware purchase is required: all development and testing will leverage cloud-based CI environments and existing OpenRefine infrastructure servers.

Funding Sources

Present: There are no concurrent cash co-funders for this NLNet application; however, we will leverage the established OpenRefine community roadmap and in-kind contributions (CI hosting, volunteer tester time) to maximize impact at minimal additional cost.

Previous funding sources for OpenRefine are available at https://openrefine.org/funding

Milestones

Milestone 1: Identify and annotate extension points
- Duration: 1 month
- Estimated effort: 90h
- Budget: EUR 5,400

Milestone 2: Improve extension isolation in Butterfly
- Duration: 3 months

- Estimated effort: 280h
- Budget: EUR 17,300

Milestone 3: Automated API compatibility check
- Duration: 2 months
- Estimated effort: 220h
- Budget: EUR 13,400

Milestone 4: Outreach and support for extension developers
- Duration: 1 month
- Estimated effort: 60h
- Budget: EUR 3,300

Compare your own project with existing or historical efforts. (4000 characters)

OpenRefine's extension ecosystem has been actively developed for 15 years. Over that time, there have been discussions about approaching the problem both from within OpenRefine itself and from within extensions themselves.

The efforts to improve the extension ecosystem from within OpenRefine seem to have largely addressed specific issues, like exposing testing utilities for extension developers (https://github.com/OpenRefine/OpenRefine/issues/6556) or resolving issues around shared dependencies (https://github.com/OpenRefine/OpenRefine/issues/6475).

There has also been recent work towards annotating parts of the codebase as explicitly part of the public OpenRefine API. This was started for the frontend in this pull request:
https://github.com/OpenRefine/OpenRefine/pull/7296
This project aims to assist this frontend work while also bringing the notion of annotating extension points to cover the backend code as well.

There has also been much discussion about moving on to an entirely new framework for extensions. The OSGi and PF4J frameworks have been mentioned as potential alternatives to the homegrown framework OpenRefine uses, with OSGi being notable in that it supports the Eclipse extension ecosystem. While this project does not explicitly include evaluation of these frameworks, it will leave the extension ecosystem in a better place to evaluate these alternatives as a future option.

Core OpenRefine developers have also contributed to automated testing within extensions themselves. Most prominent is the frontend test suite implemented in the CommonsExtension:
https://github.com/OpenRefine/CommonsExtension/pull/157
This project will extend this work by documenting the approach taken by this extension to allow extension developers to adopt such testing for their own extensions.

What are significant technical **challenges** you expect to solve during the project, if any? (5000 characters)

The OpenRefine community has been discussing the question of extension ecosystem improvements for years. Over that time, the discussion has coalesced around a few high-level areas of work. These areas include: documenting the existing extension points; navigating releases containing breaking changes; and making it easier to preserve API stability. In some cases, concrete approaches have been outlined in forum threads and GitHub issues. The milestones in this project have been designed to support the work in each of these areas. For areas that have less outlined material, the following work aims to provide clarity on a path forward.

Milestone 1: Identify and annotate extension points

The first milestone in this project will be to categorize the existing extension points in OpenRefine. While there are published guidelines for extending OpenRefine, there is a large area for potential improvement. A report on which areas of the codebase are used in popular extensions will help the community decide which they would like to support moving forward. Additionally, this stage of the project includes codebase documentation for each of the extension points found in this stage. There is an existing strategy for frontend extension points that leverages JSDoc annotations. The backend will need a similar strategy. In each case, the documentation style will need to support marking extension points as deprecated.

Deliverable:

- **Audit report**: complete JSDoc pass (issue #7017) and publish a machine-readable registry of *supported* versus *deprecated* hooks.

Milestone 2: Improve extension isolation in Butterfly

Butterfly is the web framework that OpenRefine uses to configure backend functionality. It is also how extensions can register new functionality within OpenRefine by leveraging extension points. At a high level, Butterfly allows for extensions by loading extension code from a specific directory and running each extensions' initialization script. Butterfly is no longer used by other active projects, and even OpenRefine has forked the project to customize some of its behavior.

This milestone aims to address two existing extension isolation issues: shared dependencies and initialization failures. The existing infrastructure for loading libraries on which OpenRefine and its extensions rely will make all dependencies available to the codebase. In practice, this creates a difficult situation for any extension that wishes to bring its own dependencies. One extension can introduce a conflicting dependency (either directly or indirectly) which will lead to issues across the entire OpenRefine suite. This work has been discussed in this GitHub issue: https://github.com/OpenRefine/simile-butterfly/issues/15.
The other isolation issue is that an extension can encounter an issue and leave the entire software suite in a broken state.

Deliverable:

- Resolve dependency isolation issue in Butterfly, OpenRefine’s web framework
- Minimize impact of initialization failures in extensions

Milestone 3: Automated API compatibility check

API compatibility has not been systematically tracked. This milestone will deliver a tool that can be used to automatically report on any changes in the supported external API from one version of OpenRefine to the next. This tooling can be used to reduce the maintenance burden of OpenRefine's developer community when evaluating a release for potentially impactful changes.

Deliverable:

- **Continuous compatibility tests**: GitHub Actions matrix that checks for API compatibility compared to the latest stable version, surfacing breakage early (#7362)

Milestone 4: Outreach and support for extension developers

Breaking changes can only be avoided for so long. This milestone will deliver support for releasing breaking changes through the form of automated testing for extensions that can be done to identify potential compatibility issues. Should there not be enough time for a fully automated test suite, documentation of existing test methods will provide guidance to extension developers. More so than other milestones, this will include extensive outreach to the developer community to ensure that this tooling provides an overall benefit to the extension ecosystem.

Deliverable

- Documentation of the testing work done in CommonsExtension: https://github.com/OpenRefine/CommonsExtension/pull/157
- If possible: an automated test suite for extension developers to extend

We will host **two virtual workshops** with EU and non-EU extension authors to review design memos, gather feedback on breaking-change timelines and migration tooling.

After these milestones are completed, OpenRefine will be well positioned for follow-up exploratory work.

Describe the **ecosystem** of the project, and how you will engage with relevant actors and promote the outcomes? (2500 characters)

There are currently more than 20 extensions listed on the OpenRefine extensions page, and many of them have maintainers who are active members of the OpenRefine forum https://forum.openrefine.org/, which also serves as our developer discussion list. This forum also includes users of and contributors to the core software. The forum will be the primary way of engaging with the community while research is ongoing. Project members have held open office hour calls in the past and will do so during this work as well. We will present the project's results at our annual BarCamp conference.

We will engage with our European partner organization, NFDI and HBZ, who allocate staff time to pilot testing, user feedback sessions, and documentation review. They will also co-host workshops or webinars in Germany to disseminate project results.

In addition to these standard methods of communication, additional outreach will be made in the form of surveys to the developer community, discovery calls with individual extension developers, webinars and other group calls to present and discuss the different areas of work.

This outreach will also be targeted at non-extension developers as well. While this work is primarily intended to support core OpenRefine maintainers and existing extension developers, an effect of the improvements in this project will make extension development a less daunting task for newcomers.

In the last 12 months, OpenRefine averaged 15,500 downloads per month, with 22 active contributors during this time. A total of 163 issues were created, and 157 were closed via 212 pull requests. OpenRefine receives about 800 academic citations per year.