Institut de Genetique Humaine — CNRS

OMERO
Background Import

PROJECT OVERVIEW DOCUMENT

Abstract. Background Import is an extension of the Open Microscopy Environment software platform that will allow to import images into the OMERO image repository without requiring experimenters to be logged onto the acquisition workstation, thus reducing costs and optimising resource usage of microscope facilities. This document provides an overview of the Background Import project by summarising key information about goals and scope, requirements, solution architecture, and project plan.

NB This document is an **early draft** and is subject to change.

Volker Baecker
Andrea Falconi
Julio Mateos-Langerak

June, 2015

1. Introduction

1.1. **Background and Context.** Montpellier RIO Imaging (MRI) is a multi-site imaging core facility with over 700 active users and more than 60 acquisition systems. Due to the sheer size and complexity of the structure, the staff is facing an increasing number of data management and analysis challenges. To overcome these challenges, MRI decided to adopt OMERO[1, 3] as the main infrastructure for the storage, management, and analysis of image data. There are also plans to participate in the development of OMERO to better serve the specific needs of the local imaging community.

OMERO software has been deployed to most acquisition workstations within MRI so that after images have been produced by a microscope, experimenters can import them directly from an acquisition workstation into the OMERO image repository. Of course, they still have an option to copy the image data to some other storage facility of their own choice. Either way, data will have to be transferred out of the workstation as acquired images are periodically deleted from local storage to make room for new ones. Whichever way experimenters choose to transfer the data currently requires them to be logged on the acquisition workstation until the image data have been fully transferred out of the workstation into the designated storage location.

This arrangement has the following disadvantages:

- *Lessened microscope availability.* A microscope is not available to other users until the experimenter has logged out of the acquisition workstation so the next user will have to wait until the transfer is finished.
- *Additional costs.* Because experimenters are billed for the amount of time they are logged on an acquisition workstation, the time it takes to transfer the data is billed too.

Note that these issues are not inherent to the OMERO import workflow or, more generally, to the data transfer mechanism per se, but rather are caused by the fact that the *data transfer is tied to a user session.*

1.2. **Proposed Solution.** This project aims to remove the above limitations. The goals are cost and resource usage optimisation within microscope facilities as well as demonstrating that this can easily be accomplished using OMERO technology.

To this end, we propose an extension to the Open Microscopy Environment software platform[4]. The idea is to develop a set of software components and integrate them into the existing OMERO platform so to allow experimenters to log out just after triggering an image import into the OMERO repository. The import will then run autonomously to completion, thus making the microscope immediately available to the next user and avoiding billing for the data transfer.

1.3. **Document Outline.** The reminder of this document provides information about key project areas at a level of detail suitable to support initial project execution. More in-depth and refined documentation will be provided during the course of the project as the need arises.

The next section is devoted to the software architecture. In it, requirements are stated and key solution aspects, from design to testing to deployment, are described by means of interlocking views and their relation to requirements. This information forms the basis for scoping and planning which are discussed in the following section, the Project Plan. Then, in the Methodology section, we consider software process issues, development activities and tools. The final sections explore possible avenues for future work and collect some initial considerations about the project.

2. Software Architecture

Each of the coming subsections provides an architectural view[8, 5], addressing a specific facet of the Background Import; taken together, these views convey just enough information to describe the architecture of the proposed solution at a very-high level.[1] The reader is assumed to be familiar with the OMERO platform and its implementation at MRI—deployment, operation, network topology, etc.

2.1. **Requirements.** As already discussed in the introduction, the key requirement is to untie the image import from the login session. Whatever the mechanism to achieve this, the transfer quality, in terms of reliability and performance, should not degrade. Resource usage (CPU, I/O, etc.) on the acquisition workstation should not be any higher than that of the OMERO importer and microscope operation should not be affected. Additionally, robust failure handling and recovering has to be in place so that users and system administrators are notified of failures and failed imports can be retried or resumed as appropriate; successful imports have to be notified to the users who requested them. Data integrity must be preserved and it should not be possible to remove a file while it is being imported.

2.2. **Functional View.** The idea is to have a new background process, the Import Proxy, that can run an image import on behalf of OMERO clients; this process would run outside of any user session so to untie the image data transfer from login sessions.

In detail, the Import Proxy is a Java ICE servant exposing an asynchronous call that accepts the exact same inputs as those the OMERO client currently passes to the OMERO server to perform an import (object hierarchy, annotations, etc.) except for the image data which are instead replaced by a pointer to the actual image file. The Import Proxy runs the import in the exact same way as the OMERO Java client does presently—i.e. save inputs to OMERO server, parse and transfer image file. The proxy process needs to have access to the files to import which are therefore expected to be available either from the local file system, if the client and

[1]A more detailed software architecture document may be provided during the course of the project if the need for it arises.

the proxy run on the same machine, or via a network share, if the proxy is on a different machine. Files will be locked for the entire duration of the import in order to prevent users from accidentally deleting them before import completion; further data integrity is already enforced by the current OMERO import functionality—e.g. SHA checksums. Upon successful completion, the Import Proxy sends an email notification to the user who requested the import; if the import fails, both the user and system administrator are notified. A write-ahead log provides support for recovering from failures so that broken imports may be fixed, e.g. by retrying them if at all possible or by requesting the system administrator's intervention.

Note that because the Import Proxy runs an import in the exact same way as the current OMERO client does, transfer quality and resource usage should be the same as what they presently are (i.e. no degradation) if the proxy and the client both run on the same host.

In order to be able to use the Import Proxy, the OMERO Java client is extended with a new Import Plugin. This plugin is configurable so that Background Import functionality will not be available for sites that do not require it—this is the default configuration setting. The plugin provides a new button that users, after going through the usual import workflow, can press to trigger a background import in which case the plugin calls the Importer Proxy and returns immediately; the user can then quit the OMERO client and log out while the import is run in the background.

2.3. **Development.** The Background Import will be part of the OMERO Java client (Insight) project. All the Background Import code will be developed in the OMERO Java client code base. Two separate modules will be added: one for the Import Proxy and one for the Import Plugin. Build and configuration management are already in place for the entire OMERO Java client code base, so the current infrastructure will be adopted wholesale and existing coding standards will be adhered to. Automated testing is available too, so unit and integration tests will be developed and added to the current automated test suite. The Background Import byte-code is also packaged inside the existing OMERO Java client install bundle.

Note that sharing of project infrastructure and deployment is advantageous in terms of speed of delivery but is not necessarily the best arrangement in terms of release management and deployment dependencies. So this may change in the future and the Background Import could become a separate OMERO project.

2.4. **Deployment.** In our deployment configuration, the OMERO Java client is installed on each acquisition workstation. The client is configured to use the Import Proxy which runs as a service on the same workstation. The service is set up using a control script included in the client install bundle; the script starts (stops/restarts) a Java process to run the Import Proxy code which is also part of the same client bundle. Thus, only one installation is needed to run both the client and the proxy. However, whereas the client is available to all workstation users, the proxy is a single background process running on the workstation to service all users. The proxy uses whichever OMERO server the client is configured to use. Run-time, hardware, and network requirements for our deployment configuration are the same as those of a standard OMERO Java client deployment.

2.5. **Operation.** The installation procedure entails installing an OMERO Java client on each acquisition workstation (as currently done), configuring it to use the Import Proxy, and setting up the Import Proxy to run as a service using the provided control script. Additionally, the account under which the proxy service runs needs to have enough permissions to read any image produced by the microscope attached to the workstation. Upgrade procedures for the Background Import components are exactly the same as for an OMERO Java client deployment.

The proxy service should be monitored (e.g. added to the list of processes monitored by system administrators) to ensure its availability. If the service is down, any user (of that workstation) trying to import an image would receive an error and would not be able to import. Even though this can be remedied in minutes just by manually restarting the service, it is best to try and prevent errors by automated monitoring and recovery in order to raise the quality of service. Any failure encountered by the Import Proxy while running an import will be emailed to a configured administrator mailbox which system administrators should make sure to have access to.

User training and documentation will be provided. For support and assistance, users can contact system administrators.

3. Project Plan

Below is a high-level project plan. The plan is broken down into subsequent phases (loosely following the Unified Process, see e.g. [6]) that will be executed sequentially; within each phase there is a summary of the tasks to carry out to reach the completion of that phase. Tasks within each phase may be carried out concurrently as needed. Milestones track project progress by specifying what is the expected outcome at a point in time and how long it is expected to take—i.e. a time estimate for the completion of all the tasks leading up to that milestone.[2]

3.1. **Inception.**

Goals. Initiate project and prepare ground for development.

Tasks. Establish vision, scope and justification for project. Gather key requirements and identify risks. Familiarise with OMERO platform. Build development, test, and server virtual machines.

[2]Each task was estimated separately using PERT three-point estimation: Call b, w, l the best, worst, likely task completion time, respectively; we took the expected completion time t to be: $t = \frac{b+4l+w}{6}$. Then we summed up expected completion times for the tasks leading up to that milestone.

Milestone (M1). All inception tasks completed. Virtual machines ready for development cycles. Project approved by CNRS steering committee. *Expected Completion Time*: 15 days.

3.2. **Elaboration.**

Goals. Establish and validate solution architecture.

Tasks. Analyse requirements. Study candidate solutions. Attend OMERO conference to discuss plans and development options. Select solution architecture and produce project plan. Write project overview document (only draft needed). Familiarise with ISO certification procedure and Redmine project management.

Milestone (M2). All elaboration tasks completed. Project managed in Redmine and ready to be audited. *Expected Completion Time*: 24 days.

3.3. **Construction.**

Goals. Implement system features.

Tasks. Develop initial prototype: Import Proxy ICE interface; ICE servant and client; Import Plugin button and asynchronous call to the proxy. Develop unit and integration tests. Have small group of end users test and evaluate product; steer further development according to received feedback.

Milestone (M3). Prototype is stable and other system features can be added. *Expected Completion Time*: 16 days.

Tasks. Evolve prototype into deliverable product by implementing: write-ahead log, locking, notifications, configuration, server control interface, control script. Develop unit and integration tests. Write developer documentation. Build pre-production environment to deploy and test deliverable. End to end testing. Load testing.

Milestone (M4). All construction tasks completed. All system features implemented. System fully tested and ready for deployment. *Expected Completion Time*: 27 days.

3.4. **Transition.**

Goals. Deploy system to production environment. Refine product to incorporate any feedback received from end users.

Tasks. Conduct pilot deployment using small group of end users to test and evaluate product in production environment: install OMERO Java client on selected machines, set up Import Proxy service, assign permissions, configure admin mailbox. Monitor running system and fix any issues. Collect feedback from end users and refine product as needed. Prepare end-user training materials. Write end-user documentation.

Milestone (M5). System is running smoothly in a controlled production environment. Any issues have been fixed and end-user feedback incorporated into product. System is ready for deployment to all MRI target machines. *Expected Completion Time*: 19 days.

Tasks. Deploy system to all MRI target machines. Monitor deployment and fix any issues.

Milestone (M6). All transition tasks completed. System is running smoothly in production environment. Project ends and product enters maintenance phase. *Expected Completion Time*: 5 days.

4. Methodology

We are going to adopt the Kanban method (see e.g. [2]) for software development as this is the approach the OMERO team has embraced. The project will be managed through the Redmine[7] project management tool. Source code and documentation will be kept in our GitHub repository at:

- ☐ https://github.com/c0c0n3/openmicroscopy

The development tool chain is the same as that used by OMERO core developers.

5. Future Work

The Background Import project is, at this stage, an MRI custom extension to the OMERO platform and, as such, not officially endorsed by OME. However, our use cases may turn out to be fairly common at other facilities worldwide, in which case Background Import functionality could be integrated into mainstream OMERO. We will engage with the OMERO core developers to discuss integration. This may require some changes to the Background Import code, for example, factoring it out from the OMERO Java client into its own OMERO project. Also, new features could easily be added to make it more flexible and, hence, usable in a wider array of scenarios. Specifically, the Import Proxy could operate in "pull" mode too: images could be fetched from a remote workstation; thus several acquisition workstations could share the same Import Proxy service. We estimate this to require roughly between one and two months of development.

More generally, one could envision a system in which end users need not worry or be aware of where images are physically stored or how they can be accessed. At acquisition time, an user would select where the image belongs in a given *logical* hierarchical structure of their choice—a virtual file system; under the hood the system would stream the image data to physical cloud storage. (Note that physical storage and logical hierarchical structure would be *independent*.) The cloud software would then let the user access the image from any cloud-enabled device, using the same logical hierarchical structure. OMERO would be integrated in the cloud to enable image viewing, management, and analysis which could now be done from any device connected to the cloud via a web browser. We will explore this avenue further if it is of any interest to MRI.

6. Conclusions

Once implemented, the Background Import will increase availability of microscopes, as users can log out just after image acquisition, and reduce operating costs, as users will not have to pay for image data transfers. Thus we see how OMERO technology can help optimise cost and resource usage within a microscope facility by providing robust, feature-rich management of microscopy data. However, the benefits of adopting OMERO go far beyond: we see it as playing an important role in facilitating scientific discovery. In fact, image data are ultimately acquired to extract and derive information that forms the basis for scientific investigation. As data volumes grow and data formats proliferate, extracting information soon becomes extremely difficult if the data are not accessible through a uniform, efficient interface which is exactly what OMERO provides, dramatically improving the ability to acquire knowledge form the data.

References

[1] Chris Allan, Jean-Marie Burel, Josh Moore, Colin Blackburn, Melissa Linkert, Scott Loynton, Donald MacDonald, William J Moore, Carlos Neves, Andrew Patterson, et al. Omero: flexible, model-driven data management for experimental biology. *Nature methods*, 9(3):245–253, 2012.

[2] David J Anderson. *Kanban - Successful Evolutionary Change for your Technology Business*. Blue Hole Press, 2010.

[3] Open Microscopy Environment. Omero web site: www.openmicroscopy.org/site/products/omero.

[4] Open Microscopy Environment. Web site: www.openmicroscopy.org.

[5] PB Kruchten. The 4+ 1 View Model of architecture. *Software, IEEE*, 12(6):42–50, 1995.

[6] Philippe Kruchten. *The rational unified process: an introduction*. Addison-Wesley Professional, 2004.

[7] Redmine Project. Redmine web site: www.redmine.org.

[8] D. Soni, R.L. Nord, and C. Hofmeister. Software architecture in industrial applications. In *Proceedings of the 17th international conference on Software engineering*, pages 196–207. ACM New York, NY, USA, 1995.