The FileSender project: integrated software development from Proof-of-Concept code to package

Guido Aben[1], Wendy Mason[2], Paul Bonnington[3] and Louis Moresi[4]
[1] AARNet p/l, Kensington, WA 6151, Australia, guido.aben@aarnet.edu.au
[2] Monash University, VIC 3800 Australia, wendy.mason@monash.edu
[3] Monash University, VIC 3800 Australia, paul.bonnington@monash.edu
[4] Monash University, VIC 3800 Australia, louis.moresi@monash.edu

INTRODUCTION: WHAT IS THE FILESENDER PROJECT ABOUT?

The FileSender project [1], started in 2009, is an effort to build software and services to bring the transfer of large files (up to and over 2Gb) available to every researcher and scientist, regardless of their computer/network proficiency. The project comprises of an international consortium of Research and Education (R&E) networks, making FileSender more widely relevant as a service implementation. The resulting software has been available as a stable binary (version 1.0) since January 2011, and the resulting service platform has gone in production in a number of countries – including AARNet's CloudStor [2, 3, 4] service in Australia – since March 2011.

FileSender is delivered as a web-based application that allows authenticated users to securely and easily send arbitrarily large files to other users, as identified by their email address. Authentication of users is provided through SAML2, LDAP and RADIUS; FileSender is inherently prepared to be connected to an Access Federation, and has been deployed in this mode in Australia as well as nearly a dozen-odd other countries. Users who do not possess credentials allowing them to access the service can nevertheless still upload files, provided an authenticated user sends them an invitation in the form of a single-use upload 'Voucher'.

The purpose of the software is to transfer large files, not to act as a storage platform; to this end, FileSender keeps files available for download for a certain amount of time; the file is deleted after this validity time expires.

DEVELOPMENT METHODOLOGY – BEING "CRUEL TO BE KIND" TO YOUR OWN PROJECT

In developing FileSender, we were confronted with a few design and development decisions. In a number of places, we decided not to take the path of least resistance, but to rather be "cruel to be kind" to ourselves. The major ones where a path of least resistance was available but we decided against it were:

a) Team composition

Upon identification of the requirement for such a service, a decision could have been made to simply start coding; this would have minimized startup times and potentially could have lead to earlier delivery of a beta product. Instead, the long-term goal of shouldering the maintenance load with a large user base prompted us to look for international project partners. Once these were found (in the R&E networks of Norway (UNINETT) and Ireland (HEAnet), plus additional members who joined later), a comprehensive search was conducted to identify projects, software and/or designs. Only when it was established that no suitable prior R&E software project existed did we embark on coding. While this approach caused startup delays, starting out with an international cross-organisational team has helped us to avoid pitfalls associated with making assumptions that may not be universally valid - support of UTF-8 international character set compliance is a case in point.

b) Project structure

In keeping with the goal of minimizing the support load for each project member, the project was established as a full-featured open source effort, with a commitment to releasing under the BSD license, a website, a world-readable revision repository, mailing lists, an open membership policy, lightweight governance, a documentation wiki, project meetings via teleconference etc.. This made simply starting up a rather laborious task, but the rewards later on in the project were reaped as it became comparatively easy to convince new members to join the project and contribute to it.

c) Coding standards / language / packaging

The swiftest code development would have arguably have been attained had we chosen for a language like java or python; the use of a deployment framework such as django or lift would have made sense from a rapid development point of view as well. We decided against these, arguing that it was far more important that a systems administrator working for a newly interested service provider party (and by extension a potential new support-load sharing entity) could pick up the FileSender package and have it up and running within half an hour. Increasingly, system administrators value lockdown and tight package control on their production machines; the propensity to "try new stuff" and install a handful of additionally required packages has never been great and is waning further. Hence we decided to code in the framework most universally present on web servers: php. This has been a journey of discovery, but the reward of straight-forward installability has been very worthwhile. See also under d).

d) Choice of end-user client / interface

The solution with the lowest barrier of entry we could identify would have been to develop a standalone java client, or restrict ourselves to a single supported client operating system and develop a native application for that OS. Linux would have been the OS of choice for its low barrier of entry development environment.

We decided against these options however; this was mainly informed by the observation that researchers of less than expert computer proficiency tend to work from workstations on which they have limited administrative rights, and using networks that are typically fairly tightly firewalled. Installing clients is often out of the question, and protocols other than http(s) are not guaranteed to work reliably and universally. Thus, we came to the conclusion that the FileSender service needed to be deployed as a webservice. Otherwise, it was unlikely that FileSender would ever fulfill the goal of becoming a truly low barrier of entry tool with wide deployment over as many project partners as possible – which in turn is relevant to the goal of shouldering the support load with as many entities as possible. Developing a web-based service that could upload files larger than 2 GB turned out to be painfully difficult, but the reward has been very swift uptake, both among end users (1060 Australian users as per June 2011) as well as among service providers (11 international R&E network installations as per June 2011).

e) Testing

We decided early on that proper testing was paramount if we wanted the FileSender service to have both a genuine low barrier of entry as well as the lowest possible maintenance cost for operators. The FileSender project has therefore conducted both systematic unit and system testing in-house, as well as extensive field tests of alpha and beta releases, with tester pools in the 300-400 user range.

In-house testing has been conducted by first drafting accurate workflow diagrams, detailing how modules interact with one another and with the end user. Based on that workflow diagram, a 'living document' of test cases was drafted and is continually maintained as features are added. This document covers all valid entry and exit points, verifying whether the system presents the expected outcomes in all cases. These tests were conducted by hand up until the latest release; as the time of writing this submission, an automated web testing framework (Selenium [5]) has been evaluated and found fit-for-purpose. We are in the process of scripting all documented test cases and preparing them for automation.

CONCLUSION

It is undeniably harder to set up a small piece of software as a proper project, and to plan for wide deployment beyond one's own institutional borders; often it may even seem like a superfluous effort. At the same time, software that is successful rarely stays small, as more features are added by popular demand; and once an ad-hoc project grows beyond ad-hoc software size, it becomes nonlinearly harder to untangle the ad-hoc structure and design decisions made in the past. Even for the purpose of developing a piece of software as relatively small as FileSender, we have never regretted taking the more laborious route of setting up development as a fully fledged open source project with third-party deployability in mind. We are frankly convinced that the resulting system would never have been as easy to use or maintain without the discipline and international input gained from this structure, and we have no doubt that we will engage future software development challenges using the same modus operandi.

REFERENCES

1. *Home | FileSender Space | Assembla*. Available from: www.filesender.org, accessed 29 June 2011.
2. *AARNET - Services - Cloudstor*. Available from: http://www.aarnet.edu.au/services/content-delivery/cloudstor.aspx, accessed 29 June 2011.
3. *CloudStor*. Available from: https://cloudstor.aarnet.edu.au, accessed 29 June 2011.
4. Mason, W. G., Aben, G., Meijer, J. J., Richter, C., Bonnington, C. P., Moresi, L. & Betts, P. G., 2010. *Facilitating Research Collaboration in the Australian Geoscience Community using CloudStor*. 2010 Sixth IEEE International Conference on e-Science: 106 - 112. doi: 10.1109/eScience.2010.31
5. *Selenium web application testing system*. Available from: http://seleniumhq.org/, accessed 29 June 2011

ABOUT THE AUTHORS

Guido Aben: Guido Aben is AARNet's director of eResearch. He's involved with challenges ranging from developing networking specials with particular research groups, to enticing business to deliver their products in specially packaged or licensed forms to turn them into feasible tools for research, and the occasional stint at software development. Guido has interests in international collaboration between NRENs, as well as collaboration and partnerships between NRENs and market players. He holds an MSc in Physics from Utrecht University.

Wendy Mason: Wendy has been completing a PhD in Computational Geodynamics and eResearch, in the School of Geosciences at Monash University. She has several years experience in writing user documentation and software beta / release testing for open source software development projects.

Paul Bonnington: Prof. Paul Bonnington, Director of the Monash e-Research Centre, is a member of the Go8 Digital Futures group, and on the Board of Directors for the Victorian Partnership for Advanced Computing. He is a member of the steering committees for the Victorian Life Sciences Computing Initiative (VLSCI), Victorian e-Research Strategic Initiative (VeRSI) and National Computational Infrastructure's Specialist Facility for Imaging and Visualisation (MASSIVE). Paul is also a member of CSIRO's e-Research Council, and currently serving on the National Research Infrastructure Roadmap Expert Working Group for e-Research. He recently served as the Chair of the Steering Committee for the Australian National Data Service Establishment Project. Paul is a Professor in association with the School of Mathematical Sciences at Monash University.

Louis Moresi: Louis Moresi is a professor of geophysics and computational mathematics at Monash university. He is interested in the deformation and failure of geological materials and develops novel computational techniques in this area. He oversees the development of the Underworld parallel, particle-in-cell finite element code, and has a keen interest in techniques for interacting with large-scale simulations running remotely.