Lessons learned in developing digital preservation tools the right way (and the wrong way)

Paul Wheatley
Paul Wheatley Consulting Limited
Leeds
West Yorkshire
@prwheatley
paulrobertwheatley@gmail.com

ABSTRACT

The digital preservation community has had a chequered history in developing software tools to perform operations essential for preserving digital data. Poor technology choices, half measures in adopting open source approaches, insufficient engagement with users, finite project funding and an array of other challenges have hampered tool development. Gaps in capability are common but even where tools have been created to take on a particular problem, they often face patchy support and an uncertain future. Lessons have however been learned from mistakes that have been made in the past. User engagement and an agile development approach can focus solutions on real problems. Adoption and expansion of existing tools (sometimes from outside of this community) can yield greater and more dependable results. Focused designs can make adoption easier. Sharing the experimentation and data behind tool development and assessment can be as invaluable as the tools themselves. This paper provides an outline of lessons learned from developing digital preservation tools across JISC and EU funded digital preservation projects, such as PLANETS and SCAPE and more recently from agile hackathon and mashup events run by SPRUCE.

Keywords

Digital Preservation, User Requirements, Digital Preservation Tools, Open source development

1. WHY CAN'T WE HAVE TOOLS THAT JUST WORK?

At iPRES2012 Steve Knight noted dissatisfaction in our preservation tools in a review of the previous decade of digital preservation: “Tools like DROID and PRONOM etc. didn’t work properly then, and they still don’t work properly now. The wish list from this year’s Future Perfect Conference (New Zealand) did not differ that much from the wish list four or ten years ago.” [1] Even tools that have (at least previously) seen considerable use within the community, such as JHOVE, are facing an uncertain future [2]. There are an array of reasons behind this:

- Many community created tools have struggled to survive once grant funding ended
- Tools from outside of the digital preservation community have often been overlooked
- Users have had insufficient say in the focus and design of preservation tools resulting in a mismatch between genuine user needs and preservation capabilities
- Organisations have in theory adopted open source development approaches, but this has often gone no further than depositing software in a code repository at the end of a project
- Little thought has been given to how new tools will be integrated with existing workflows
- A lack of even basic testing has resulted in a painful installation experience for many users

As Johan van der Knijff put it: “Why can't we have digital preservation tools that just work?” [3]

Lessons have been learned from the last 15 years of digital preservation research and development. Most of the unwieldy applications developed by the PLANETS Project [4] have not seen significant uptake, but focused preservation tools such as Jpylyzer [5], developed by SCAPE [6], have already been adopted by various organisations and have been embedded in workflow tools such as Goobi [7]. Engagement with existing tools that offer much to the preservation community, such as Apache Preflight [8], has yielded significant improvements to the tool. Pairing developers with practitioners in SPRUCE Mashups [9] has highlighted the rapid progress that can be made in solving preservation problems in even a short space of time (sometimes resulting in new preservation tools). These new approaches represent a sea change in the community which is beginning to focus more on practical experimentation, to share and exchange ideas with others using social media and to engage more readily with existing open source projects.

2. LESSONS LEARNED IN DEVELOPING PRESERVATION TOOLS

The poster will present the following lessons learned in developing preservation tools, and is adapted significantly from the SPRUCE Mashup Manifesto [10]. This in turn is based upon experiences in exploring, and in some cases solving, over 150 specific preservation challenges [11].

- **Be agile in your first steps**
  - Develop/prototype in short bursts, then demo and get feedback from your practitioner/user
  - If you don’t achieve results within a few hours, you are probably doing it wrong. Try a different approach
  - Get crude results quickly, perfect and polish later
  - Scripting languages can be useful for delivering quick results
- **Re-use, don't re-invent the wheel**
  - Most problems have already been solved, although often not by the preservation community

- Experiment with existing solutions first
- Someone else will have experience of other tools to try. Twitter can make the connections. Check the COPTR tools registry [12]
- Re-use existing code before writing any of your own. Existing code comes with existing users (who test and report bugs) and existing contributors. Exploit this where possible!

- **Keep it small, keep it simple**
  - Functional preservation tools should be atomic
  - Modularise in the face of growing requirements
  - Think about how someone else will integrate your tool in a workflow. Make it easy for Preservica, Rosetta, Archivematica and the rest to incorporate your code
- **Make it easy to use, build on, re-purpose and ultimately, maintain**
  - Test driven development simplifies subsequent maintenance
  - Share your source
  - Automate your build
  - Package for easy install
- **Share outputs, exchange knowledge, learn from each other**
  - Write up your experiences and share them (sharing less than successful experiences is just as valuable as successful ones!)
  - Publish the data you generate. This tool->this data->these results
  - Shout about it, blog it, tweet it, and add a tool registry entry to COPTR

Boxouts and examples will be used to expand on key points from the above.

3. REFERENCES

[1] Steve Knight quote in: Angevarre, Inge, NCDD Blog. http://www.ncdd.nl/blog/?p=3338

[2] Gary McGath's blog. http://fileformats.wordpress.com/2014/03/08/statejhove/

[3] Johan van der Knijff's blog. http://www.openplanetsfoundation.org/blogs/2014-01-31-why-cant-we-have-digital-preservation-tools-just-work

[4] Planets Project website. http://www.planets-project.eu/

[5] Jpylyzer, JP2 validation tool. http://openplanets.github.io/jpylyzer/

[6] Scape Project. http://www.scape-project.eu/

[7] Goobi, digitisation workflow management tool. http://www.digiverso.com/en/products/goobi

[8] Preflight, PDF validation library, http://pdfbox.apache.org/cookbook/pdfavalidation.html

[9] Paul Wheatley and Maureen Pennock. Supporting practical preservation work and making it sustainable with SPRUCE. *iPRES 2013 proceedings*. http://purl.pt/24107/1/iPres2013_PDF/Supporting%20practical%20preservation%20work%20and%20making%20it%20sustainable%20with%20SPRUCE.pdf

[10] SPRUCE Mashup Manifesto. http://wiki.opf-labs.org/display/SPR/The+SPRUCE+Mashup+Manifesto

[11] Digital Preservation and Data Curation Requirements and Solutions, http://wiki.opf-labs.org/display/REQ/Digital+Preservation+and+Data+Curation+Requirements+and+Solutions

[12] Community Owned digital Preservation Tool Registry (COPTR), http://coptr.digipres.org/