Code Foster Parenting

Bringing up someone else's FOSS

Ben Roberts (ben.roberts@nesi.org.nz)

Outline

The general problem

It all comes down to economics. Raising good code is a costly exercise.

Solution one: Proprietary software

- Developed and maintained in the course of a business.
- Licensed to users for a fee (one-off or ongoing).
- Developers typically few and paid.
- Ongoing maintenance usually depends on ongoing profitability.

Solution two: Open-source software

- Developed and maintained by a user community.
- Usually gratis, though support may be sold separately.
- Developers typically volunteers, or software development is incidental to their duties. May be few or many.
- Ongoing maintenance depends on ongoing goodwill.

The specific problem

Orphaned code.

The Moscow Orphanage. Photo by A. Savin.

Zonation

- Conservation planning software
- Developed by the Statistical Ecology group at the University of Helsinki (https://www.helsinki.fi/en/researchgroups/metapopulation-research-centre/software)
- Hosted on GitHub (https://github.com/cbig/zonation-core)
- Released as open source under the GNU GPL version 3
- Released as source code and as pre-compiled binary
- **The original team has moved on. The code is orphaned.**

Department of Conservation PMR team

- Divide conservation estate up into Management Units (MUs)
- MUs differ in size, terrain, number of vulnerable ecosystems, etc.
- Need to prioritise MUs
- Use Zonation, R, etc. to help them do this

Mahuika

- NeSI capacity cluster, replacing Pan
- Different operating system and software stack to Pan
- Zonation precompiled binary worked on Pan, not on Mahuika

NeSI Computational Science team

- Partner with scientists to achieve computational research outcomes
- Parallelisation, optimisation, compilation...

The goal

Deploy Zonation 4.0.0 on Mahuika.

Configuring and building

- Zonation uses CMake. This wasn't a problem.
- Various dependencies: GCC, Qt4, Boost, zlib...
- Some of these were challenging.

Boost

- A collection of C++ libraries (www.boost.org)
- Prone to substantial change over time.
- The most recent version known to work with Zonation was 1.55.0.
- We tried 1.69.0 (the then production release) and 1.55.0. Neither worked.
- Wrong number and/or type of arguments...
- Perhaps Boost itself was only part of the problem?

Qt4

- Used for various purposes, including GUIs. Zonation requires it for some reason.
- Each build of Qt4 relies on a specific version of Boost.
- We used the system Qt4. This will have been built against the system Boost.
- Now we're getting somewhere...

Cracking the problem

- We don't want to build the entire X11 display system.
- Therefore, we need to use the system Qt4.
- Therefore, we need to use the system Boost.
- It turns out we also need to use the system GCC. Zonation, and these old Boost and Qt4, were written using an old C/C++ standard, no longer available in recent GCC releases.
- **We managed, in the end, to build and deploy Zonation.**

Code Maintenance

- Code needs to work with all its dependencies.
- This means updating to reflect language standards, API changes, and so forth.
- Containerisation may be an option, keeping a stable virtual system for a while.
- But is any given container a medium- to long-term solution, or a short- to medium-term workaround?

Conclusions

- Open-source software needs committed parents in order to grow and flourish
- If your business depends on an open-source package, you might have to adopt it, or at least become part of its development community while it still has one.
- This will need a plan, a business case, a sponsor who recognises its importance...
- Are you ready?

Choices

- Adopt the orphaned package
- Switch to another package with a stronger "family", maybe a proprietary package
- Decide that the benefits of that line of work aren't worth the costs and risks

Questions & Answers