Technical Approaches to Improved Software Sustainability

Presented at CW3S19, Collegeville, MN

Mark C Miller

July 24, 2019

LLNL-PRES-XXXXXX

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Outline

- Practices that help improve a project's "sustainability trajectory"
- Continuous Technology Refreshment (CTR)
- Some random examples of good SQE Practices for Libraries

What is Continuous Technology Refreshment (CTR)

The periodic upgrade or replacement of infrastructure to deliver continued reliability, improved speed, capacity, and/or new features.

- IT world uses a hardware-centric view of CTR to define
  - Processes and policies (typically driven by costs including competitiveness)
  - System components (storage centers, networks, servers, desktops, mobile)
  - Frequency of update of components
  - Rolling updates throughout the system

- bssw.io blog post

Examples of CTR Activities

- Build system
  - Gmake →GNU Autotools→Cmake→Spack
- 3rd party dependency major versions
  - Python 2→3, MPI 1→2→3
- Scripting
  - Tcl→Bash→Python→Julia
- Issue tracking
  - Cq→Redmine→GitHub
- Revision control system
  - ClearCase→Subversion→GitHub
- Testing dashboard
  - Ad-hoc→Cdash→Bamboo
- Continuous Integration testing
  - Travis→CircleCI→Azure
- Planning tools
  - Ad-hoc→MS Project→Kanban
- C++ language standards
  - 9x→11→17
- Re-licensing (Spack example)

What Drives CTR Activities?

- Technology obsolescence
- Expanding development team / processes
- Access larger toolbox / new features
- Loss of resources
- Performance and/or Quality improvements

CTR work not always possible to plan for

Often, CTR improves developer's lives but not user's

Recent CTR Experiences for VisIt 3.0 Release (bssw blog post)

- Binary content in repo
  - Non-support→Git LFS
  - Gotcha: Forked repos
- Revision control system
  - Subversion→GitHub
  - ClearCase→Subversion (2006)
- Issue tracking
  - Redmine→GitHub
  - ClearQuest→Redmine (2006)
- Documentation
  - OpenOffice→Sphinx+RTD
- 3rd Party Libraries and Tools
  - VTK: 6→8
  - OpenGL: 2.x→3.x
  - HDF5: 1.8→1.10 (deferred)
  - gzip→7z for data tarballs (~2x smaller)
- Branching and merging model
  - Old: BrA→RC→Mainline
  - New: BrA→RC, BrB(Cp-BrA)→Develop
- Build system (2006)
  - Autotools→Cmake
  - build_visit shell script

Differences in SQE Standards for Libraries vs. Applications

- Users of a library care about how it is compiled and installed
  - Gcc vs. icc vs. pgcc
  - Optimized vs. debug
  - Serial vs. MPI Parallel (which MPI)
  - Static vs. dynamic
  - All above extend recursively to any 3rd party dependencies
- API changes break consumers
- Necessity to support (bugfix) older versions
- Documentation much more technically rich and detailed
  - What it does and how it works

```
/* high-level API: compresse
/* open compressed stream an
zfp_stream*         /* alloc
zfp_stream_open(
  bitstream* stream /* bit s
);
/* close and deallocate comp
void
zfp_stream_close(
  zfp_stream* stream /* comp
);
/* high-level API: compresse
/* bit stream associated wit
bitstream*                  /
zfp_stream_bit_stream(
  const zfp_stream* stream /
);
```

```
/* high-level API: compressed stream construction/destruction --------
/* open compressed stream and associate with bit stream */
DEF_FUNC(
zfp_stream*,          /* allocated compressed stream */
zfp_stream_open,(
  bitstream* stream /* bit stream to read from and write to (may be NU
))
/* close and deallocate compressed stream (does not affect bit stream)
DEF_FUNC(
void,
zfp_stream_close,(
  zfp_stream* stream /* compressed stream */
))
/* high-level API: compressed stream inspectors ----------------------
/* bit stream associated with compressed stream */
DEF_FUNC(
bitstream*,                  /* bit stream associated with compressed s
zfp_stream_bit_stream,(
  const zfp_stream* stream /* compressed stream */
))
```

Some random examples of useful practices for libraries

- Giv… version number meaning in terms of
- Pr… co…
  - #i…
  - #e…
  - #e…
- En… co…
- In a polymorphic languages, fake it
  - …lse”
  - …ds
  - …pers
  - …talled

	A	B	C
	Major Digit	Minor Digit[a]	Patch Digit
In the worst case, changes in this digit *can* mean...	*Everything Minor means &...* Major API changes Major feature enhancements Major file format changes[a]	*Everything Patch means &...* Minor API changes Minor feature enhancements Minor file format changes[b] Performance improvements[e]	Documentation updates Bug fixes API additions Performance improvements[d]
impact on application when digit changes	re-type <= impact <= re-think	rebuild <= impact <= re-type	none <= impact <= rebuild
typical frequency[c]	years	months	weeks

a. Another common practice is an odd/even minor digit to indicate development/production releases.
b. File format issues are specific to I/O libraries. d. High-impact/low-cost. e. Lower-impact/higher-cost.
c. Our experience has been that increment of the release number is often triggered at regular intervals by routine bug-fix work while increment of minor and major number is triggered as planned development activities are completed.

Some random examples of useful practices for libraries

- Never abort or throw an uncaught exception in a production build
- Encode messages of API symbol deprecation in warnings emitted from the package itself

What about sustainability of data as opposed to code?

- Material databases and equations of state play a crucial role in simulation results
- CAD models and their discretization for input to simulations
- Number representations
- Self-describing file formats
- Reproducibility of results

Don’t forget about the ”us” in sUStainable

- Our enthusiasm for our work
- Our vision and body health
- Beware of cognitive overload
- Default one hour meetings
- More effective communication
- Fine grained multi-tasking is inefficient and mentally draining
- When I am tired and when I rush…I make more mistakes…which cost more to fix
- Foster safe and inclusive work environments for each other


Lawrence Livermore
National Laboratory