Rivet: past, present and future

Andy Buckley
University of Edinburgh

7th MCnet meeting, CERN, 2010-01-13

Contents

Review

Quick summary

- Rivet is (to 1st order) HZTOOL++
- Tool for replicating experimental analyses for MC generators
- With some lessons learnt:
  - big emphasis on generator independence
  - split steering from analysis
- Tools and key analyses in one system
- Also: usable as library or executable, dev or user
- Current release: 1.1.3, from June 2009

Some characteristics

- Using HepMC pipes for generic processing with **`rivet`**
- LWH implementation of AIDA for histogramming: long overdue replacement
- All analyses loaded from shared libraries as "plugins"
- Reference data bundled. . . most exported from HepData
- Code structured with "projections" to cache computations
  - Once used to the idiosyncracy of "applying projections" to the event, this makes writing analyses very clean and compact
- "AGILe" gen interfaces for convenience with Fortran gens $\rightarrow$ HepMC
- Main source of Professor input

Version 1.2.0

Recent developments

A lot of work since June. . . no such thing as a minor release!

- Rewritten (much nicer) plugin system
- Rewritten (neater and more efficient) projection lifetime/dependency system
- *Minimal* hard-coded normalisations: scaling in post-processing
- Improved run control means energy known at `init` time: really allows big clean-ups. . . unexpected bonus!
- Metadata as external YAML files for analyses, plot config files for histos, etc.
- HTML, PDF and interactive documentation from metadata
- **Lots of min bias (UA1, UA5, STAR, etc.) analyses, general projection improvements, bug fixes. . . physics!**

Validation

1.2.0 is a(nother) big improvement. . . release currently blocked by quality control.

Need to systematically validate performance of $\mathcal{O}(80+)$ analyses. For every release. Or every change. . .

Val frameworks awkward to fully automate: structured collection of scripts/fragments seems the way forward. **But needs work *now*.**

Generally, we want something that eliminates (i.e. automates) this by-now very painful step, so we know asap when something functional has been broken. Integrate with existing Hudson continuous build checks: functionality as well as compilation checking.

Getting critical

Pre-mortem

Rivet design and development has changed a lot since we started:

- design is cleaner, more focused on analysis and gen independence
- development is more systematic
- increasing emphasis on quality control and documentation

So which ideas have been good, and which have been stinkers?

Getting it right is crucial to making Rivet a long-term HEP community success.

Successes

- autotools, SVN, Trac, HepForge, etc.
- Boost, YAML,
- Small tools doing one thing well (cf. UNIX philosophy)
- HepMC pipes – use system abstraction features
- Documentation via metadata
- Python/C++ hybrid design (compile $\rightarrow$ runtime deps), and UI emphasis in both API and CLI
- " $\sim$MVC": usable as lib or executable
- HepData connection. . . eventually!
- Systematic validation: emphasis on quality control, segregation of UNVALIDATED
- HepMC/fastjet/MC/observables development & scrutiny: but HepMC not followed by experiments
- Hudson builds: extend to validation

~~Failures~~^W Sub-optimal features

- Histogramming: LWH and YODA inertia
- Systematic validation: haven't delivered (yet). Painful!
- Lack of HERA analysis migration – see next point!
- Community-building: have users, but few external contributors: certainly with expts, but also patchy in MC community
- Manpower: we have a problem...IMO, need experiment dev input

And probably many more that I've forgotten. Core dev/management focus needs to be on these points.

Conclusions

Summary

I'm biased, but I think Rivet has turned out as a really nice tool. Nice reception at Les Houches, lots of positive user reports, LHC experiment uptake...and I/we have learnt a lot along the way.

Lots of lessons learnt – some/most of the nicest design features weren't part of the original picture! Rewrites & migrations are painful, but have made it a better tool. Try to minimise incompatible changes, but change takes time.

Rivet 1.2.0 will be another milestone: *validation and emphasis on quality control are paramount*. **This requires work.**

"Final" reworking is histogramming: need an expressive, simple and portable histo library which supports run merging. Plans are all there, just needs manpower.

Where is main Rivet (etc.) core development going to come from in coming years?

Thank you!

A huge thank you to all the MCnet and otherwise-associated people who've contributed to Rivet over the last couple of years:

Hendrik Hoeth, Frank Siegert, Holger Schulz, James Monk, Gavin Hesketh, Eike von Seggern, Emily Nurse, Lars Sonnenschein, Peter Richardson, Christophe Vaillant, and more.

Please keep contributing!