PPTAP Computing & Software Roadmap Workshop, 19th July 2021

Software

Luke Kreczko

University of BRISTOL

CMS

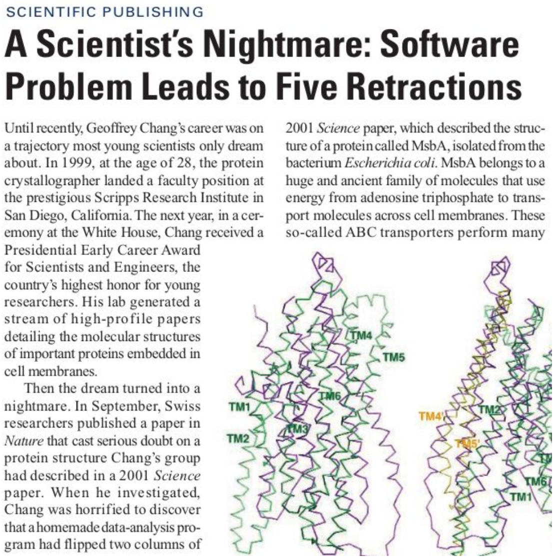
DUNE DEEP UNDERGROUND NEUTRINO EXPERIMENT

LZ

SWIFT-HEP

Why do we care about software?

Science requires reliable and reproducible results


TM4
TM5
TM6
TM1
TM2
TM3
TM4'
TM2'
TM5'
TM6'
TM1'

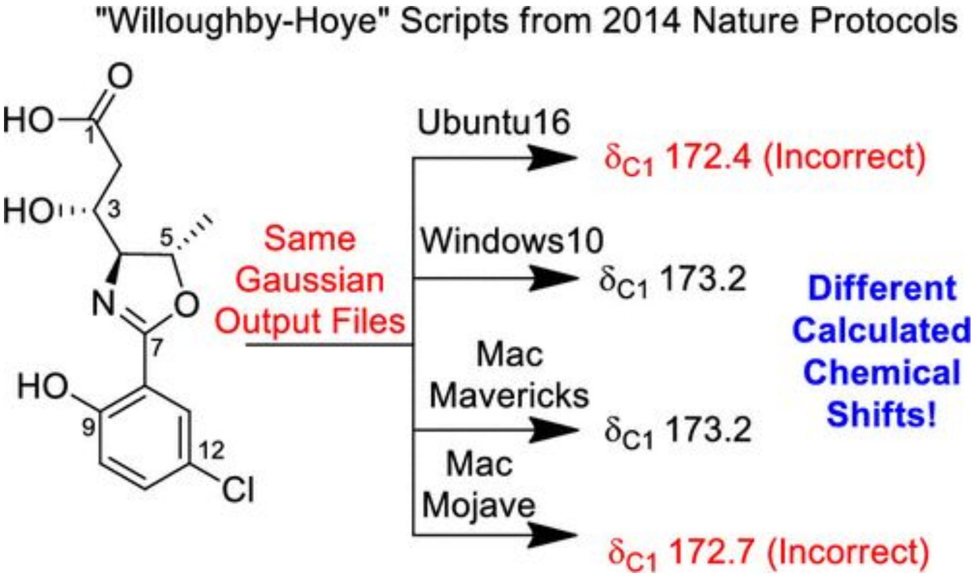
SCIENTIFIC PUBLISHING

A Scientist's Nightmare: Software Problem Leads to Five Retractions

Until recently, Geoffrey Chang's career was on a trajectory most young scientists only dream about. In 1999, at the age of 28, the protein crystallographer landed a faculty position at the prestigious Scripps Research Institute in San Diego, California. The next year, in a ceremony at the White House, Chang received a Presidential Early Career Award for Scientists and Engineers, the country's highest honor for young researchers. His lab generated a stream of high-profile papers detailing the molecular structures of important proteins embedded in cell membranes.

Then the dream turned into a nightmare. In September, Swiss researchers published a paper in *Nature* that cast serious doubt on a protein structure Chang's group had described in a 2001 *Science* paper. When he investigated, Chang was horrified to discover that a homemade data-analysis program had flipped two columns of

2001 *Science* paper, which described the structure of a protein called MsbA, isolated from the bacterium *Escherichia coli*. MsbA belongs to a huge and ancient family of molecules that use energy from adenosine triphosphate to transport molecules across cell membranes. These so-called ABC transporters perform many



"Willoughby-Hoye" Scripts from 2014 Nature Protocols

Same Gaussian Output Files

Ubuntu16 → δ_{C1} 172.4 (Incorrect)

Windows10 → δ_{C1} 173.2

Mac Mavericks → δ_{C1} 173.2

Mac Mojave → δ_{C1} 172.7 (Incorrect)

Different Calculated Chemical Shifts!

December 2006:
https://science.sciencemag.org/content/314/5807/1856

> homemade data-analysis program had flipped two columns of data

8th October 2019:
https://pubs.acs.org/doi/10.1021/acs.orglett.9b03216

> Python script returned files in different order depending on OS

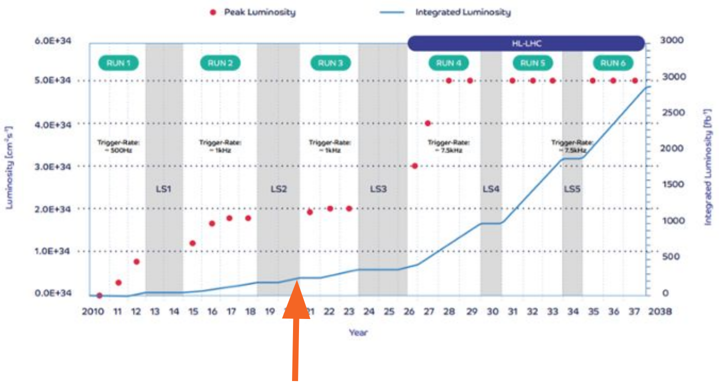
Software in HEP

- long chain of software is used between data recording and final findings: recording, processing, managing data & simulation
- Collaborations will usually provide a software framework
    - Big collaborations can afford software engineers for crucial parts
    - smaller collaborations might rely entirely on researchers as developers
- Analysis software is more diverse: each analysis area might have its own solution
    - Software development skills can differ significantly between individuals

Current challenges

- Software needs to be maintained beyond one PhD generation
  - Big experiments will safeguard their data for > 20 years
  - Can today's results be reproduced in 20 years?
- Students and researchers are rarely trained in software engineering practises
  - Unit-tests, Continuous Integration, validation of releases → need a path for sustainable software development
- Distributed resources (cluster, Computing Grid) have an entry barrier
  - time is needed to learn the system, even more time when things go wrong
- When software goes wrong it takes time away from research

Upcoming challenges in a nutshell: a lot more data



We're here (delays due to pandemic)

Computing resources are not expected to scale with data

- Need better algorithms, strategic placements of hardware accelerators
- reduce failures in software and computing infrastructure → save computing and researchers' time

Software needs to be portable and efficient on highly distributed computing infrastructure

- A lot of expertise required to make this happen → Researchers, research software engineers (RSEs) and computing infrastructure experts would need to collaborate

How can we face these challenges?

Not starting from scratch

The UK is involved in international software efforts

- We have a presence in the HEP Software Foundation, scikit-HEP, PyHEP (2 conveners)

SSI with its fellowship programme nurtures advocates for good software practices

Distributed computing and storage expertise (GridPP)

UK founded (U. Bristol, RAL) FAST-HEP effort investigated new approaches to analysis software that reduce code needed → expertise and interest is present

Just started: SWIFT-HEP to cover a wide range of Software & Computing R&D:

- Bringing researchers, RSEs and computing infrastructure experts together
- R&D in Computing infrastructure, event generators, simulation, reconstruction & trigger, and analysis

Software and computing R&D

Extend projects like SWIFT-HEP - only with serious personpower we contribute to solutions

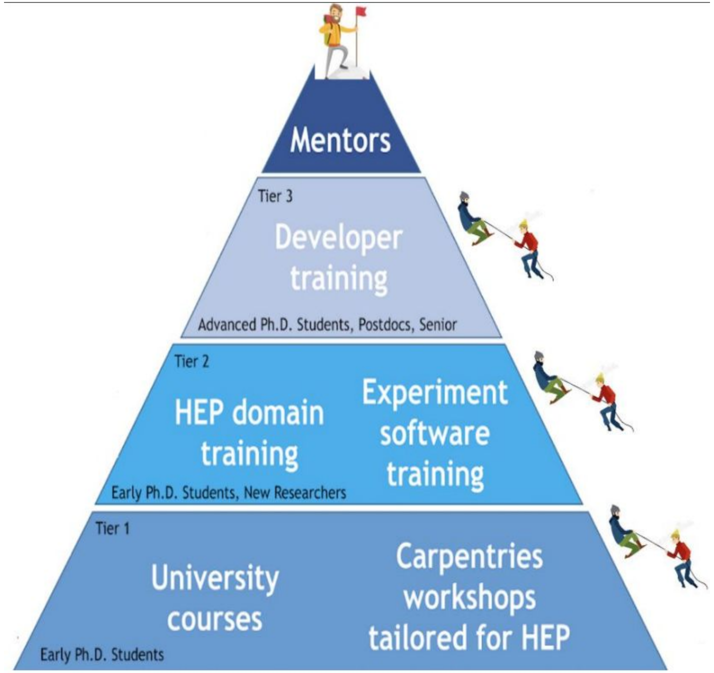
- Improve shared software → target areas that need attention most (e.g. event generators, simulation)
- Encourage collaboration between computing infrastructure and software projects

Is there a way we can establish HEP specific RSE teams within the UK institutes?

- E.g. within local groups to adapt software to latest R&D improvement, but fraction of their time within UK HEP RSE pool for joint and international efforts

Should we make sure that new experiments ensure funds for dedicated software development effort?

Training


Mentors
Tier 3
Developer training
Advanced Ph.D. Students, Postdocs, Senior
Tier 2
HEP domain training
Experiment software training
Early Ph.D. Students, New Researchers
Tier 1
University courses
Carpentries workshops tailored for HEP
Early Ph.D. Students

HSF Training vision (link)

Good training opportunities exist in the international community

- Need to make sure we collaborate to avoid duplication

We’ve started additional courses for PhD students

- Mostly on programming languages → need more on software engineering

Software development skills are also sought outside of research

- What can we do at undergraduate level?

Summary

Big challenges are coming - need to increase expertise

Software is a key to successful research - investments in training and infrastructure paramount

Existing and new approaches/technologies need to be explored in a **collaboration** between computing infrastructure and software projects

Backup slides

Declarative analysis

Data processing “simplified”

This approach seems to be welcomed by current generation of PhD students

Easy to get started, only touching code when adding new algorithms

Shifts a lot of “How to implement analysis” to “What I want to be done” → easy to share analysis procedure, more reproducibility

This disconnect allows experts to improve software “behind the scene” (e.g. for portability or caching)

Jupyter notebooks

Interactive Analysis on distributed resources

These kinds of workflows seem really desirable by the current generation of PhD students - a quick way to explore data and prototype algorithms

Shifts a lot of “How to do distributed computing” to “What I want to be done” → declarative approaches are great for research

This disconnect allows experts to improve computing infrastructure “behind the scene”, e.g. data access and pre-processing

A.I. for writing code

Good, bad, ugly consequences?

Note: Google slides version contains GIF

With the rise of GPT 3 first products appear for writing code (Github co-pilot)

Implications can be wide-ranging

- Less time spend on coding
- A.I. can suggest buggy/non-optimal code
- Who owns code written by A.I?

Software and computing R&D

For analysis: use declarative approaches to reduce entry barriers for researchers and create gateway for research software engineers (RSEs) to improve software and computing infrastructure under the hood.

Extend projects like SWIFT-HEP - only with serious personpower we can make an impact on international efforts

- Increase contributions to shared software → target areas that need attention most (e.g. event generators, simulation)
- Encourage collaboration between computing infrastructure and software projects

Create paths for new experiments to fund dedicated software development effort

Outline

- Software in HEP & challenges
- Current and upcoming challenges
- How can we face these challenges