Morning session:

Names:
Matthew Feickert
Amir Farbin
David Lange
Daniel S. Katz
A. Aurisano
Jim Kowalkowski
Justin M. Wozniak
P. Calafiura
Sally Seidel
Sumanth Mannam
Ilija Vukotic
Carlos Maltzahn

S2I2 HEP/CS Workshop Questions

Please write your ideas here for discussion questions for the Thursday sessions. (Including your name is optional.)

Is novelty necessary for computer scientists to work with HEP or can it be just an engineering collaboration?

What are the distinct software domains/communities in HEP? (e.g. Distributed Computing, Core Computing, Tracking, Calorimetry, Machine Learning, Analysis) What are their problems? Can we describe this problems in a way that can be comprehended by and is appealing to CSist?

What software components will have to be retained? What components can be rewritten from scratch? How do we decide? When do decisions need to be made? Are there different types of software for which decisions need to be made at different times?

Should we formally train HEP students in software engineering practices (Agile, Scrum)?

- (see https://www.youtube.com/watch?v=oyLBGkS5ICk for an interesting very recent talk and thoughts about how large collaborations and software dependencies can work better than they do today)

Should we run HEP software development like a software company?

How much should we rely on industry offerings?

Should we make a distinction between CS/HEP collaboration aimed at developing something novel, versus collaboration aimed at engineering a solution to a problem?

How does HEP approach CS professionals vs CS researchers? HEP probably needs to work with both, but both are looking for different things from the collaboration (e.g. and simplistically, salary vs papers)

Are there fundamentally different ideas that could completely change how HEP software is developed, that would involved bringing together relatively new or non-well-known CS techniques into HEP? (for example, automated code generation from physics algorithms that would be provable correct and future-proof)

How do the Scientists and Engineers work on gathering requirements?
Does the innovations in CS influence the HEP innovations? (can it generate new HEP problems)?
How to encourage CS people to work for HEP problems?
Is there any agile methodology to work across different contents and timezones?

There is a lot of domain-specific vocabulary and language used within the HEP and CS communities. How will they come together to understand each other and what the needs actually are?

What are examples of successful CS-HEP collaborations, and what properties have driven their success?

How to align the CS research mechanisms (3 year grants, student developers, conference pubs) with the longer term needs of big science (30 year projects, production software, journal publications)?

How to engage a broader slice of the CS community and make scientific computing more respectable within CS circles? (A commonly heard complaint in CS: scientific computing is a “niche" research area.)

What CS technologies, techniques, and trends could the HEP community adopt, rather than doing everything internally? (Keeping in mind the long time scales and production needs of HEP.)

Could HEP describe some long-term challenges that don’t need to be solved immediately, but that CS people could go off and think about? (D. Katz)

What CS research challenges exist within HEP where CS researchers could contribute to HEP but also receive recognition for their work in the CS community? (D. Katz)

How could an HEP software institute facilitate interactions between the CS and HEP communities?

What are the incentives for such collaboration for HEP people? For CS people? For non-CS people? E.g. recognition, funding, publications, students, new problems to solve, new places to apply technologies, new solutions to current problems, pride in working on a global-scale problem.

How can we create “crystallization points”, shared artifacts that allow the encoding of tools and practices of the two communities and that can be improved over time? (Successful examples are wikipedia, linux kernel, docker registry) (C. Maltzahn)

- Along these lines DIANA HEP (in particular Kyle Cranmer and Lukas Heinrich) is working on some of this in the form of preserving analyses with use of Docker (c.f. RECAST). Though Docker has some problems with HPC envs(?) (M. Feickert) This article has useful pointers to efforts making software containers viable for HPC (C. Maltzahn).

Re: data "privatization" in Frank's presentation (commentary here from R. Gardner):

- "Public" and "private" have different meanings in collaborations. In Frank's talk "public" meant datasets available collaboration-wide, e.g. public to the collaboration, and private meaning the end-stage datasets specific to an analysis and not necessarily registered in the experiment's official catalogs, even after publication (Frank correct me if this is wrong).
- The implication of this if true is that it prevents full reproducibility of a published result; there's a little "black hole" of data (and potentially software) between the collaboration datasets and the final plots and figures data.
- In the future we want "published" results to come with published data, and software, allowing for reproducibility by future analysts.
- How can CS help here? (many projects out there - are they addressing problems relevant to the scale and timeframe of HL-LHC?)
    - Check out the Popper convention -- this effort views reproducibility as a software engineering problem (the dev/ops community has already sophisticated tools to reproduce behaviors in a continually evolving software artifacts) and is partly funded by the Big Weather Web NSF SI2-SSI project. It’s a convention, not a particular tool set (although tools need to be “scriptable”). So it should be applicable to a wide variety of domains. It’s also scalable because it uses git for provenance and the git repositories include large resources by reference. (C. Maltzahn)
- (D. Katz: see https://mpsopendata.crc.nd.edu for some work in this area)

What role common data formats play in fostering collaboration with computer scientists. I.e. moving away from ROOT formats to open formats, those used e.g. in other data driven sciences? (R. Gardner)

How can CS help build frameworks/organization/processes that incubate software from the S2I2 into open source projects? Do organizations like AMPLab – UC Berkeley which build tools that have strong industry-coupling and support apply? How do we avoid building HEP unicorns, but technologies that are potentially of broad interest and with large, open development communities? (R. Gardner)

- Check out Center for Research in Open Source Software (CROSS) at UC Santa Cruz. The research project portfolio is currently skewed towards storage systems but the goal is to create a career path for Ph.D. students to become open-source software leaders. The membership agreement and the bylaws are strongly inspired by NSF's U/ICRC concept. (C. Maltzahn)
- Red Hat also provides great resources for open source in education. (C. Maltzahn)

What open source tools supported by industry can the HEP community use to solve its problems? Some good examples are OpenStack and LLVM (Spark, Tensorflow, also various commercial "AI as service" offerings, see IBM Watson for example), are there more out there? (L. Sexton-Kennedy)

What are software domains/communities in HEP? E.g. Distributed Computing,

Which of the many specialities in CS is most useful for HEP? (consider: machine learning, software engineering, computer vision, programming languages, networks, databases, complexity theory, robotics, human computer interaction, systems, architecture, ...) (N Ernst)

- This isn't an answer in full, but some examples of where applications of the above are currently being used in HEP (M. Feickert)
    - Machine Learning (DOI's and e-Prints): 10.1038/ncomms5308, arXiv:1609.00607, arXiv:1612.01551
    - Machine Learning and (some) Computer Vision: 10.1088/1748-0221/11/09/P09001, arXiv:1611.05531
    - Programming languages (incomplete, others please add): C++ (ATHENA, ROOT), Python (PyROOT, applications of scikit-learn)

Do we have a software architecture for the HL-LHC software? If not, how will we get to one? (D. Katz)

What would such an HL-LHC software architecture cover? Everything? Facilities, Operations, physics (sim, production, analysis)?

What is trustworthy software?
How to balance trust and security? (C. Maltzahn)

How do you maintain trust in software in a large organization without being bogged down? Do you rely on communities (“eyeballs”), formal methods (e.g. automatic provers), reuse mechanisms such as (certified) software containers? Who is trusted for what? What does “adult supervision” look like?

What role can software containers play in HEP software? Can they help us interface with external software - particularly new machine learning systems?

What are the venues for communication between the CS and HEP communities? (Workshops, tutorials, etc.)

Are there different levels of interactions depending on what part of the overall computing problem is being tackled? When is direct interaction with physicist-researchers needed? When is interactions with laboratory operations most important? When is interactions with software development staff at laboratories most interesting?

How does the HEP community adapt new concepts and changes, given the goals to produce physics, especially when experiments are in operations? Is it always a long-term process? How does that relate to mechanisms of reuse and leveraging the work of communities outside the HEP community?

There is big gap between downloading software and successful deployment of software -- what mechanisms should the HEP community use to bridge this gap, using automation and reuse? (C. Maltzahn)

How do we achieve consensus on software system (or product) changes across organizations for things that are shared, especially when requirements (including here organizational constraints) diverge or differ? Is consensus necessary?

Should the HEP community connect with the DevOps community? (C. Maltzahn)

The gap between practical software development / engineering (needed by the physicist) and the computer science research that required means to be a problem. How is this gap narrowed? In others words, how is the CS research coupled to applied software needs of an experiment in HEP?

What topics are useful research topics in both HEP and CS?

How is HEP software development investment measured (what is the metric)?

How do we get datasets to open communities so they can participate in challenges on regular basis in order to rapidly grow knowledge bases of new technologies or techniques?

How do you assess the value of software used for HEP research? How do computer scientists, software engineers, IT experts, and professional consultants help increase returns on that value or decrease costs?

What are the relevant computer science domains and what are their respective incentives to collaborate to bring value?

How do you leverage industry for the application challenges in which CS theorists do not find research value, but the HEP community could benefit from deeper knowledge bases?

Many computer scientists are not interested in application, what scientific value do highly generalized (and successful) machine learning algorithms such as Deep Learning or Random Forest have to computer scientific researchers?