Training Chapter - Community White Paper

June 2017

Contents

1 Introduction

Throughout the last few decades, the field of HEP has consistently either driven or utilized the latest innovations in computational tools and technologies for problem solving. The tool-set of the particle physicist is vast and ever-growing, and the problem set increasingly complex. A typical graduate student or postdoc working on HEP experiments will likely encounter a number of tools, software and programming languages with which they are totally unfamiliar. This requires "on-the-job" training for the same people whose work drives the reach of our analyses in parallel as they develop the tools that will be implemented for them.

A strong research community should consist of independent actors who ask research questions and direct their own analyses, instead of blindly following prescriptions and recipes. It is therefore imperative that the community be proactive about providing training resources and opportunities for the good of the community, and that we follow through on the promise that adequate career path possibilities exist for people in HEP who leave the field due to limited advancement opportunities.

To produce the best possible science, it is important for the broad HEP community to invest in training physicists in essential software and software engineering skills, which exist at many different levels. The level of depth of software engineering skills in which a physicist should be trained will vary depending on the problems that they face, and the appropriate level may evolve over time according to the problem at hand. For example, for a researcher who only performs basic data analyses, it may be sufficient to be familiar with just Python and data analysis libraries. However, contributing to a large open source project, such as ROOT, requires greater training in software development and related practices. These higher-order skills may include an understanding of writing testable code, time and space complexity implications, and an understanding of common software architecture paradigms. The requirement of these skills cannot be fulfilled completely just by hiring specialsits in computer science, since a thorough understanding of the physics problems is usually very important to the success of the effort. Thus the necessity of providing a career path for those young physicists willing to steer their interests in that particular direction.

In all HSF workshops the need of proper tutoring, training and relevant career possibilities in HEP computing have always emerged as felt as crucial and essential by almost all participants. There is wide consensus that future software devlopments will enormously benefit from a successful implementation of such training programs, provided that proper career prospects are put in place as an incentive to would-be trainers.

This document tries to address the most relevant questions related to training in the HEP community, namely:

- how to incentivise members of the HEP community to train students and/or other collaborators?
- how to properly assign credit to software development (and training thereof)?
- what are the current hiring practicies in the field for hiring researchers with a strong specialization in computing and what are the policies to retain shuch expertise in the long-term? Is there room for improvements?
- how to address the gap in formal training that is not always provided by the universities?

2 Motivation

With the introduction of increasingly complex tools, we have driven increasing improvements in the performance of our analyses. It is the analyzers, however who drive these advancements through novel implementations of the technology. In this context the term *analyzers* includes all those researcher who participate to a HEP experiment, from the desing and implementation of a high speed DAQ system, the setting up of a modern database to store the state of a detector, through the complex task of managing the data flow from the first trigger level to the final n-tuple dataset, up to the final task of creating sophysticated statistical and mathematical tools to extract publishable physical meaning from those same data.

Thus, the development of new tools is insufficient to drive advancements in the field without researchers who have the expertise to apply them to physics questions. Furthermore, a larger, better-connected community with expertise in the tools and access to new developments can emulate the success of the open-source model in HEP. The efforts to train the community in all relevant tools is, thus, a priority within the R&D effort in itself.

However, as tools become more complex the learning rate of developers must keep up with newly developed techniques if we are to utilize them effectively. Simultaneously, as in every branch of research, it is imperative to preserve efforts to maintain reproducibility, which requires expertise on previously used tools to be maintained. This, again, highlights the need of physicts appropriately trained in computing science, rather than professional computing scientists, who have generally goals and interests not necessarily strongly focused on the physics results of a particular HEP experiment.

To avoid duplicate efforts within the community, it is important to find a way to approach training efforts holistically and cross-experimentally whenever possible. This requires a certain level of organization and coordination in order to guarantee the delivery of efficient, non redundant software to different experiments and to allow the different communities to agree on common standards and procedures.

Beyond the immediate benefit to the scientific community in having well-trained collaborators, most people who start a physics graduate program will have careers outside of academia. This is reflected in several calls for proposals from organizations like the National Science Foundation (NSF) or H2020 promoting training in computer infrastructure and information areas.

> *The overarching goal of these programs is to prepare, nurture and grow the national scientific workforce for creating, utilizing, and supporting advanced cyberinfrastructure (CI) that enables cutting-edge science and engineering and contributes to the Nation's overall economic competiveness and security.*
>
> **- Training-based Workforce Development for Advanced Cyberinfrastructure (CyberTraining) Webinar**[1]

> *New information, communication, and computational technologies have had profound impacts on the practice of science (in this solicitation, the term science includes the natural, mathematical, computing, and social sciences), engineering, and education. This includes the means by which citizens of all ages use science and engineering to enhance professional and private lives. The systems, tools, and services emerging from these new technologies are linked to create a comprehensive cyberinfrastructure that is enabling individuals, groups, and organizations to advance research and education in ways that revolutionize who can participate, what they can do, and how they do it. Sustaining this revolution across all areas of science, engineering, and education requires the formation of a citizenry and workforce with the knowledge and skills needed to design and deploy as well as adopt and apply these cyber-based systems, tools and services over the long-term. The opportunity for such preparation should be available at all stages of formal and informal education (K-16 and lifelong), training and professional development, and must be extended to all individuals and communities.*
>
> **- Cyberinfrastructure Training, Education, Advancement, and Mentoring for Our 21st Century Workforce (CI-TEAM)**[2]

An open issue, though, is still the kind of incentive that our community is able to leverage in order to increase the number of people (with strong expertise in computing) who decide to dedicate time and effort to train fellow members of their own community. To be effective, training must be done by people who work at the cutting edge of the technology, and, usually, these are essentially found among the youngest researchers. These are the same people who are most in need of officially recognized credits for the advancement of their careers: today young scientists only receives credit and recognition (in the vast majority of cases) by working on a data analysis project and this prevents them to find time and incentive to transfer their acquired know-how to others in an effective way. A possible solution to this conundrum could be the establishment of specif career paths for researcher with a computer science specialization. The practical implementation of such an approach is made difficult, though, by the wide range of institutions involved, each with his own policies, belonging to different countries with different founding strategies. An important point to consider is the conceptual difference between the figures of a *"computer scientist"* and that of a *"physicist reseracher with computing science specialisation"*. The two cathegories have different goals, seek different paths to the solution of their problems and usually do not even adopt the same jargon in expressing problems and solutions. This makes the development of specific solutions difficult and cumbersome: in an ideal world physicsts should make a detailed plan of what they need (expressed in a tidy requirements document) and computer scientist should use this document to provide the required solution, with very little additional exchange of information during the process (this is how it works in industry). In the real HEP world, most of the time the requirements are bound to change for a variety of reasons (not only because of a poor planning) forcing developers and physicsts to closely interact during the entire process of software development, and this works smoothly only if both communities have the same goals and speak the same jargon. Finally, the career of these two figures follows slightly different paths, with different requirements.

3 Training needs of the community

The HEP user community consists of a diverse set of users; in experience level, interests, and available time to learn new techniques. If we view the entire community as virtually enrolling in a community-wide software

[1] https://www.nsf.gov/events/event_summ.jsp?cntn_id=190179&org=NSF
[2] https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=12782

training course, an undergraduate doing a summer research project has different needs and abilities than their mentoring professor. Any sort of training program must take into account these target audiences. For purposes of training, it is useful to think of all our user community as *students*. To be more specific, we identify four broad sets of students, classified by experience level, recognizing that even within these groups there will be people who will learn at different speeds and with differing depths of understanding.

- *Beginners* - New collaborators with no knowledge of the tools or techniques. These might be undergrads but they could also be senior researchers who have not yet engaged in any software analysis. These are people in need of some kind of formal training in modern computing techniques (basically compiled and scripting languages together with operating system basics such as file-systems and command-line shells);
- *Intermediate users* - HEP analyzers with some experience in either analysis concepts and/or tools, but are looking to supplement their experiences.
- *Advanced users* - HEP analysis experts who have mastered current (at that point in time) concepts and implementations and want to stay up-to-date with new developments.
- *Software developers* - HEP scientist in charge of software development in areas not strictly related to just analysis, such as DAQ systems, computing infrastructure (grid, cloud and the likes), databases, pattern recongition and complete frameworks

Where appropriate, any training program should take advantage of developments in pedagogy, such as active learning[3], peer learning[4] or web-based training (see the appropriate paragraph in the *Initiatives for future training programs* section below). In some cases, it may even be advantageous to have code samples that are purposely broken or flawed, and ask students to fix or improve them. Learning the material in such a way that it sticks with the students is difficult and is a challenge for both the students and the instructor and often takes more time than we would prefer. However, it is the best way to ensure an educated community that can fully contribute to the physics programs at large, which is really the ultimate goal of any training program.

3.1 Knowledge that needs to be transferred

At all stages of *software & computing training*, we should take care to encourage Good Practices Across the Community (GPAC), such as error checking, modularity of code design, writing tests, etc. All the key concepts addressed in the training should not be specific to a particular experiment or field of application, but general enough to be useful for the whole HEP community and possibly beyond. In this section, we present a list of specific concepts that need to be taught to members of the community, in order to guarantee the base level of competence needed to write efficient code for the different tasks performed in HEP experiments.

Base knowledge to be transferred includes basic programming concepts, data structures, basics of code design, error checking, code management tools, validation and debugging tools. More advanced topics include modularity of code design, advanced data structures, evaluation metrics, and writing tests. and working with different types of hardware accelerators. Finally, cutting-edge approaches such as the use of hardware accelerators (whether GPU or FPGA's), Deep Neural Networks or Machine Learning should be considered important training arguments. Special emphasis should be made on reporting results and documenting them. Here is a loosely collected list of training subjects that the community considers important to pursue:

- Basic and Advanced Programming Concepts
  - Object oriented paradigm
  - Compiled langauges (C++)
  - Scripting languages (Python, Javascript,...)
- Algorithmics
  - Boost library
  - STL algorithms for containers
  - R and/or ROOT
- Existing frameworks (development or application level)
  - Qt
  - ROOT
  - experiment specific framework (possibily if of potential interest outside the native experiment)
- Code design (design patterns)
- Development tools
  - IDEs (Integrated Development Environment)

[3] http://www.crlt.umich.edu/tstrategies/tsal
[4] https://en.wikipedia.org/wiki/Peer_learning

    - Debuggers
    - Profilers
- Evaluation metrics
- "Trust" metrics such as data driven tests
- Specific software implementation training
- Good practices
- Code style and clarity
- Scripting and data cleaning
- Reporting results reproducibly
- Writing Documentation

4 Challenges

There are a lot of experiment-specific training efforts. But we have some common needs. We should probably strive to extract that common knowledge and build common training from that, because it enables us to duplicate less effort on experiment-specific training, and to do the shared training better by accumulating more expertise into it.

Within a single experiment, different skill sets are needed. In addition to a base skill set that contains basic programming language knowledge, testing and code management tools and experiment-specific framework knowledge, there are more specialized skills that only a subset of the community needs to know, such as software optimization, or low-level hardware interfaces.

5 Implementation Roadmap

The implementation of training should target multiple training formats and knowledge transfer: such as videos, twikis, lectures, jupyter notebooks and advanced visualizations, etc. People who learn in non-conventional ways should not be excluded from the opportunity of learning. Since there are different stages of knowledge transfer, programs should be developed that encourage experts to share knowledge. Issues considered in this section include global training initiatives, strategies for leveraging existing forums and training tool development.

An important point to consider is the difficulty, often experienced in the past, in developing large software programs across different experiment's communities. This can be reflected in an analogous difficulty in organizing a Working Group for training across experiments or even across diverse communities (such as HEP and Astroparticle, for example): this is a point where an appropriate incentive program might be of help.

5.1 Initiatives for future training programs

Some initiatives which can be implemented for global training include: Massive Online Open Courses, hands-on workshops, online knowledge bases, expert trainer volunteer networks and web-based training approaches.

Massive Online Open Courses (MOOCs) can be used to develop an open-source set of tutorials and tools. Existing online courses such as Udacity&Coursera can be evaluated and recommended to the community. A novel approach such a WikiToLearn could also explored to assess potential benefits (as has already been attempted in the context of the GridKa School of Computing, see below).

Experiment-specific and global knowledge bases can be established with incentives for experts to contribute. They can be open-source so that a lot of knowledge can be added by the trainees themselves as information is learned (this is the particular context where an approach such as WikiToLearn could be of great help).

Question-answer websites such as Stack Overflow for HEP are also a very useful resource for common problems and questions. This approach has already been considered by HSF start-up members, and turned out to be difficult to pursue for a number of reasons, but in the future boundary conditions might change, making this approach viable.

Hands-on workshops are an invaluable part of learning how to apply theoretical concepts in practice. Identifying which of the current workshops are productive and useful, and if they cover all the topics that need to be transferred and that are in demand by the students is an important action item.

Creating an expert tutor volunteer network is another way to provide training and support to the community. This of course requires proper recognition, at least in terms of career prospects as an incentive for young researchers. It is clear that the best possible tutors are, in principle, those people who are actively engaged in modern software developments: these are in general young researchers in the first steps of their careers and, in order to be allured to devote substantial time to training and tutoring, they must be assured that such an effort would be properly recognised in an official way. Such kind of recognition is currently not implemented, at least not in a standardized or official way.

It should be considered that some professors, who act as graduate student advisors, might need to be encouraged to make sure their students are properly trained. Sometimes, students are instead pushed to learn the bare minimum to get the work done, at the expense of a broader training/education curriculum which could actually yield improved results, but further down the line. One incentive would be to provide training programs which also count as course credit, perhaps towards an elective. This model is already in limited use with some online solutions[5], but this is not universal. Discussions should be started with collaborators at higher education institutions to see what the roadblocks or opportunities would be for these training sessions to serve double duty. It should also be considered that not all students will end up their career in reasearch or academia: their contribution to the reasearch activity, as students, should therefore also provide them knowledge, know-hopw and skills considered as a valuable asset by industry, in order to increase their chances of a career outside reasearch.

A difficulty that has emerged in the past with respect to implementation of training courses is the lack of funding along with the lack of available time by experts in the field. People with enough expertise or insight in the field have usually no time to devote to prolonged periods of student's training, and, even when they can find some, the cost of setting up a training course in an effective way is often beyond what's made available by funding agencies (funds for travel, hosting, setting up a room with a computing infrastructure to allow interactive hands-on session, etc.). A possible way out is a completely different approach to training (but complementary to the already existing and successful classical efforts such as the CERN School of Computing's Bertinoro and Kit ones): instead of directly teaching to students, trainees could make use of a web-based platform to provide training materials to students. This complementary approach has several advantages over traditional ones:

- the tutor can add material to the web site at a very slow pace (whenever he finds time to do it, one slide a week or a chapter a day)
- his material, made public on a web-site, can be further expanded by collaborators (also at their own pace) or, even better and more productively, by students who decide to contribute with new additions, examples, exercises etc. Such a collaborative effort allows more people to be exposed to training at any given time, creates a sense of community and creates bridges between people in contiguous areas or research. Students can use that same platform to exchange their own examples, make suggestions and point out interesting concepts. In such a model, the possible contribution from others to the training material needs to be moderated and validated with appropriate policies.
- if complemented by the availability of remote virtual machines (possibly through a browser), students could have access to examples and exercises that are already embedded in their own natural environment: all the necessary tools/libraries needed to implement the exercise will be already available in the virtual machine (possibly a Docker container). With just a web-browser, students could run complex examples from home, taking advantage of a remote facility that provides some storage and computing power. Important here is the concept of "environment": a Docker container could be setup in such a way that the student will work in an exact replica of the environment he/she will be exposed to in his/her experiment (or collaboraton at large). Moreover, a student could be provided with a Docker container that preserves his modified environment across sessions, allowing him to develop his skills for a prolonged period of time by accessing all the files that were made persistent day by day during the training.
- there would no longer be a need to find the resources to host a school and pay the tutor(s) (and eventually subsidize the students to participate the training in a remote location). Students could follow the training at their own pace from wherever they happen to be: a traditional school only lasts for 5 days (10 at most) and it is difficult to cover a subject to any significant depth in such a short time. The web approach, instead, would allow for very long and in-depth coverages of any kind of subject, and in this sense it could be a *complementary* approach to a traditional school. Of particular interest could be courses such as, e.g., "Machine Learning", "Statistical Analyis with ROOT", or even just "Good practices in C++ or Python Programming for scientific computing".
- Finally, this approach could allow the creation of *browsable* repositories of all training materials, grouped by argument, by relevance, by experiment or whatever other critieria. Students from all over the world could be exposed to a large repository of examples, exercises and in general training material from their own home.

Such a web-based platform already exists, and has been implemented as an Open Source project (backed by Wikimedia) by a group of italian students (the group of developers is already above 30 people). The project is WikiToLearn [6]. It hosts training material in several languages, for several disciplines, ranging from Economy, to Physics, Mathematics and several others. Being based upon wikimedia[7] software, it is very easy to add material to the site, make it appear under a specific topic (such as, e.g., Software/Techniques/Machine-Learning) and manipulate it as if it would be a single document. In the end, students can selectively choose individual chapters

[5] https://www.edx.org/credit
[6] https://it.wikitolearn.org/
[7] https://www.wikimedia.org/

from the site and have the corresponding pdf sent them as a book, complete with index, content and chapters.

The adoption of such an approach is made rather easy in WikiToLearn by the relative simplicity of the wikimedia-based toolset: users contribute their training material using just a web-browser, and in order to do this efficiently the learning curve has been kept appropriately shallow.

An interesting exercise in this context has recently been made by colleagues of the GridKa School of computing: the session of this year (2017) has been made publicly available on WTL[8]. This constitutes an intersting example of what can be accomplished using this platform: it is just a first example of what's possible, but an inspiring one.

Finally it should be important to evaluate if and to which extent complementary approaches to training (such as schools and dedicated web-site courses) could be of mutual benefit (how to make them cooperate in the development of a complete training program).

5.2 Leveraging existing forums

To achieve our goals for training the community, we can take advantage of existing training forums. Resources such as conferences, workshops, and schools (in person and online) can provide a lot of value for our training purposes with little effort to set up. We should leverage these existing training forums that are already developed that most closely match the HEP community's needs.

Within the HEP community, there are already some working examples of dedicated training environments that alternate between general topics and experiment-specific topics. The LHC Physics Center (LPC) at Fermilab hosts Hands-on Advanced Tutorial Sessions (HATS)[9] throughout the year to introduce and train participants in topics as diverse as the latest b-tagging algorithms, Git/GitHub, and Machine Learning. These HATS provide face-to-face time with instructors and participants at Fermilab, and also allow remote collaborators to join in by video and complete the same online exercises. A similar approach is in use in the CMS Data Analysis Schools (CMSDAS)[10], a series of week-long global workshops that now take place at multiple labs all over the world and are designed to ramp up new collaborators in CMS-specific analysis tools while providing some discussion of the physics as well.

Other examples are CERN School of Computing [11], CERN OpenLab Software workshops [12] education in collaboration with industry partners, and a series of more advanced topical training courses provided by MPI Munich and DESY that focus on advanced programming, use of acceleration hardware and statistical tools including machine learning. This list includes the Bertinoro [13] and the GridKa [14] Schools of Computing.

Over the past decade, massive open online courses (MOOCs) have been developed by universities and private organizations. They have been well received by industry and academia and are maintained by the outside community. In addition, they provide a lot of flexibility in terms of cost and time constraints; they are typically free and open for enrollment at any time of the year. Since the material can be accessed at any time and revisited at any time, material can be completed at a pace that makes sense for a HEP that needs to learn ML in a piecemeal way.

There is a growing number of viable options of MOOCs teaching various subjects. Since there are many options, there is a wide variety with respect to the depth of the material and specific tools taught. Exploring these options allows us to choose which is the right offering for the knowledge needed to work on a specific HEP experiment. We can pick and choose modules to tailor an appropriate road map of skills to learn. More difficult will be to assemble HEP specific training material not already avaible elsewhere in an efficient and organized way, since this requires adequate organization, volounteers and and a suitable infrastructure.

Several industry conferences already exist that bring together those in academia and industry who are at the cutting edge of these techniques. Conferences such as NIPS[15] and PyData[16] provide a focused place where attendees can learn a lot about ML in a short period of time. ML concepts such as current ML methods, tools, and problems facing industry and academia can be learned at conferences. In addition, conferences are an excellent networking opportunity; attendees can meet and share ideas with fellow ML learners and experts. Bonds can be formed quickly at conferences that can be maintained after the duration of the conference. These connections to the outside community can be essential since we will be evolving training materials to ensure that they stay relevant over time.

[8] https://en.wikitolearn.org/GridKa2017
[9] http://lpc.fnal.gov/programs/schools-workshops/
[10] http://lpc.fnal.gov/programs/schools-workshops/cmsdas.shtml
[11] https://csc.web.cern.ch/
[12] http://openlab.cern/
[13] https://web.infn.it/esc17/index.php?lang=en
[14] http://gridka-school.scc.kit.edu/2017/
[15] https://nips.cc
[16] https://pydata.org

For example, at the time of this writing, Coursera[17] and Udacity[18] both provide great machine learning massive open online courses (MOOC) at no cost. These two courses both provide a great foundation for learning ML fundamentals. However, Coursera's approach emphasizes more theory (with more math background necessary) and uses MATLAB/Octave while Udacity's approach emphasizes practical using ML techniques in a practical sense using Python tooling.

5.3 Resources and Incentives

Training the community is something that a large cross-section of the community understands is important, but finding time and effort to contribute to this effort is not actively on the radar of most potential contributors. Providing incentives for their participation and creating the appropriate platforms can go a long way to reach a productive training environment. It can be as simple as inviting someone to give a software tutorial on the subject that they are familiar with, give a lecture or seminar or contribute to a growing knowledge base.

Another important incentive is recognition. For younger members of the community, having the opportunity to create a training resource, such as a software tutorial or a knowledge base on a particular topic, is very empowering and motivational to continue the efforts of training others. Engaging younger members of the community is crucial to long-term success of HEP training endeavors.

It is also critical to incorporate training into grant proposals so that it can be connected with other areas such as research and development. Efforts like DIANA-HEP [19] and aMVA4NewPhysics [20] that combine training and software development are good examples of such ideas in practice. More such examples are needed.

Misc contributions

Software Carpentry[21] and Data Carpentry[22] are two organizations that collaboratively build and teach some of the basic concepts in developing and maintaining software, and analyzing data, respectively. The materials that they develop and use are open, and can be customized for science domains (e.g., HEP), or participant groups (e.g., undergraduates). Their model is that they offer training of trainers, and then the trainers who have graduated can offer training under the SC/DC names, though of course, anyone can use the SC/DC materials without doing it under the SC/DC names. Subjects covered in a typical Data Carpentry school are given below. This list is taken from the curriculum of the CODATA-RDA Research Data Science Summer School in progress in Trieste, Italy during July 2017. (http://indico.ictp.it/event/7974/)

- Introduction
- UNIX Shell programming (Software Carpentry Module)
- GitHub (Software Carpentry Module)
- R (Software Carpentry Module)
- BYOD (Bring Your Own Data) best practices
- Data Management with SQL (Software Carpentry Module)
- RDM Storage Management
- Visualisation
- Machine Learning Overview - Recommendation
- Recommender Systems
- Artificial Neural Networks and other Machine Learning Systems
- Research Computational Infrastructure

While Software Carpentry is leveraged to build a foundation of knowledge for later more advanced concepts, it is important to note much of this material is developed specifically for this course and not a part of the larger Software Carpentry program. This course focuses on a shallow but wide building of foundational skills approach, introducing many base concepts but not going deep into any one concept, with a through line of Open Science and Ethical Data Usage.

[17] https://www.coursera.org/learn/machine-learning
[18] https://www.udacity.com/course/intro-to-machine-learning--ud120
[19] http://diana-hep.org/
[20] https://amva4newphysics.wordpress.com/
[21] https://software-carpentry.org
[22] http://www.datacarpentry.org

6 Conclusions

The HEP community by and large aknowledges and recognizes the great importance of training in the field of scientific computing. This activity should encompass several types of *students*, from undergraduates, to young researchers up to senior physicists, all of them in need of an appropriately designed training path in order to be proficent in their scientific activity.

We identified a certain number of problems that need to be overcomed in setting up an appropriate training problem:

- Costs and relative funding
- Incentives
- Career paths
- Overall organization across experiments, countries and corresponding Funding Agencies

For each of these points we provide an overview of the current situation and possible proposals to build a road-map.