HEP Software Foundation

Input on scope, purpose, intent and organization by **INFN** members.

Disclaimer:
The following document represents the point of view of a limited number of INFN scientists and professionals working in the context of computing in HEP and Nuclear Physics. The official standing of INFN on such matters, properly signed by the management, will be produced at a later stage.

D. Menasce, on behalf of these INFN members.

Background

Scientists involved in research in Physics, in particular in Experimental High Energy Physics have become accustomed to rely for their work on the existence of a rich and varied portfolio of software suites or even complete frameworks that have been developed by different organizations over the years. Notable examples are well known, like ROOT and Geant4 in the context of frameworks and libraries, but also VOMS and CREAM in the context of Grid Middleware, but several others exist and are utilized on a wide range of purposes and uses. The examples mentioned pertain to two different domains: the first being the software components directly employed by users in the context of experiments and physics analysis, while the second domain is part of the infrastructure currently used by the experiments for the management of large amounts of data in a heterogeneous computing environment like the Grid. Both domains are crucial for the researches but the infrastructure layer (Grid, as a catch-all name) is by many taken for granted (somebody is doing it for everyone and is therefore more of interest to a specific group of professionals) while components of the first domain have a more direct influence on the development of code by end-users who therefore manifest a particular sensitivity to it.

All of these components need to evolve over time to take into account new user needs as well as advances in technology, but this evolution requires sustainable funding, coordination and an appropriate technical forum where stakeholders can exchange their different views and solutions. Moreover, the foreseen evolution must take place without disruption on the availability of existing components, taking also into account backward compatibility (at a reasonable level at least). So far all existing systems were proposed, started their development and began production without or little concern about their mutual interaction, the adherence to specific standards (e.g. for interoperability) and the establishment of appropriate evolution-schema to adapt to new technologies such as vectorization and multi-threading.

Since sustainability of the production, maintenance and development of software with adequately high levels of quality is of concern for all funding agencies, the proposed Collaboration should try to involve as many communities as possible considering the overlap of their mutual interests and expertise. Examples are, beyond HEP, the Astro-Particle, Astrophysics and Nuclear Physics communities, but contribution from others as well could be beneficial to provide a robust portfolio of software components designed by professionals, but driven by specific and solid use-cases.

An informal survey in our country has shown that a Software Collaboration, Foundation or Forum (non necessarily labeled HEP only) could, in principle and under some assumptions, be a possible instrument to help managing the needed harmonic evolution of the existing tools and packages as well as promoting the emergence of new ones.

Goals

From the meeting held at CERN on 3-4 April the message emerged that people in our community are in general sympathetic with the idea that a stronger cooperation of the stakeholders involved with the development and the maintenance of the common software we rely on can be beneficial indeed, especially on the medium and long term, but critical issues remain to be sorted out concerning the particular form in which this cooperation will be stated formally and in detail. We feel it is premature, at least in this particular document, to focus on such political aspects and prefer instead to discuss what the developer and user communities consider important for their activities and what their expectations are in this context.

In this document, for brevity, we will therefore concentrate on items not already evidenced in contributions from other fellow institutions, with which we generally agree.

We highlight, in the following, a few goals (among many possible others) that people in our institution feel important in the context of the proposed *Collaboration*:

- The primary goal should be to achieve a sort of *economy of scale*: large software projects (existing or proposed new ones) require adequate man-power, infrastructure in the form of managed repositories, continuous contact with the user community of reference, handling of a dashboard with project details and up-to-date documentation, an efficient help-desk with very quick response capabilities, tutorials and many more of those costly attributes. All existing packages, tools or frameworks have, at various levels, implemented their own infrastructure to deal with such matters. Merging these commonalities, providing a unified model for the deployments and maintenance of these packages could be of help in abating costs. A large part of the foreseen evolution of software computing technologies relies on advanced vectorization and parallelism, which is a field cutting across many of the existing tools and packages in use by our communities. Factorizing this specific domain of expertise could help developers in specific areas to concentrate on the development of their own tool, seeking advice from a selected group of professionals devoted to the most technical aspects of modern computing. This would require the establishment of a sort of open forum of experts in various domains of computing to which developers might refer for their work. Again, having just one such group (or a limited number, to preserve bio-diversity) could in principle help achieving an *economy of scale* by providing a high level of expertize with a uniform approach to all software components, avoiding the frequent syndrome of *reinventing-the-*

wheel. To this extent the collaboration should also be proactive in suggesting, whenever possible and advisable, already existing solutions to proposed new project by regular surveys of the marketplace.

- A second goal, equally important, is to foster multidisciplinarity. HEP is a prime example of the merging of a vast number of disciplines, ranging from low-energy Nuclear Physics to extremely high energy phenomena, cutting across detector technologies involving solid state physics, optics, electronics and real-time computing, just to name a few. In this scenario the involvement of expertise from the relevant domains is of paramount importance whenever possible. To make a concrete example: the Geant4 toolkit incorporates theoretical and experimental knowledge from several physics domains and, as such, its continuous development along with thorough validation requires expertise in all these contexts, which is something seldom found within a single physics community. Since it is deemed necessary for many toolkits to evolve, in particular from a sequential approach to a parallel/multi-threaded technology, the appropriate convergence of mutual expertise in both physics and computing must be accomplished. One goal of the collaboration, among others, should therefore be to promote the creation of an appropriate venue (a technical forum) where these aspects could be discussed, designed and reviewed, leaving the final development of the code in the hands of specialists of the specific domain the toolkit is supposed to address.

- One of the major barrier users find in adopting or employing software is lack of appropriate training and tutorials. Software complexity has increased over time much more rapidly than the capacity of people to learn new technologies/methodologies on their own, and plain documentation of tools like Geant4 are no longer sufficient to overcome the initial barrier to adoption. Therefore we feel that one of the important goals of the collaboration, besides coordinated exchange of ideas about the evolution of these software components, should be the establishment of a permanent training infrastructure, where people can acquaint themselves with new advances and improvements in technology: something akin to the CERN School of Computing or the INFN annual School of Bertinoro, but directed more to accomplished scientists than to young students only, who usually have more time and attitude to self-learning. New tools, even when they are exceptionally powerful and useful, are adopted slowly by the community at large just because of lack of this kind of proper tutoring. The establishment of an *ad hoc* kind of SWAT-team could be of help, a team of specialists that work together with the developers of a specific project or framework for a limited time to bring them on the right track. Funding, sustainability and policy of action of such a team is certainly a very critical and complex issue and we leave its discussion aside here, but the whole research community could certainly benefit enormously from the existence, availability and help of such an infrastructure.

- Another possible goal of the collaboration should be a technical forum, where user requirements are collected and prioritized and proper decisions are taken in order to satisfy them in an efficient way.

- An important keyword in modern computing (maybe the most important one) is interoperability, which stems from the existence and adherence to appropriate standards. There is a long list of organizations that promote standards in the field of Computing and we consider strategic the participation of members of the proposed "Collaboration" to as many such bodies as possible, in order to make the voice of our user and developer communities be heard with the necessary advance time, before critical final decisions are taken. Our participation to these activities will also have the benefit of keeping our community at the cutting edge of software evolution at a time when technical knowledge of this kind can make a difference in improving the use of our computing resources in terms of energy efficiency.

- Finally, we deem essential that all software dealt by the "Collaboration" should be based upon some variant of the Open Source Licensing scheme. This brings into focus the problem of Intellectual Property, which should be thoroughly examined and discussed: as a general principle, the IP should be in the hands of the developers or their Funding Agency, but the details are many, subtle and out of scope of this document.

Scope and Duration

Let's briefly review the panorama of existing HEP SW components:

1. Toolkits of very general use like Geant4 and ROOT. These are mature projects that may need to upgrade to new computing technologies such as vectorization/parallelism and concurrency, but also take into account new features and advantages brought forward by advances in the C++ language (C++11 yesterday, C++14 today, moving towards C++17 tomorrow). Some or all of the toolkits in this area may greatly benefit also of advances in the technology of continuous integration, code performance analysis (both static and dynamic) by sharing some of the underlining components. Also new languages and compilers are merging and adequate knowledge in these areas is also very important.
2. Middleware components developed for the GRID that need to be maintained to allow current operations for the LHC experiments (but not only), some of which could even evolve to be used also in Cloud computing environments (such as VOMS, for e.g.).
3. Packages and frameworks born within a single experiment, with potential of more widespread usage. Notable examples are RooFit, VDT, glideinWMS, Panda, AGIS, Gaudi, etc. The authors are committed to keep the software functional for the hosting experiment, but they see the potential of an enlarged user basis, and are generally keen on helping with its adoption, provided the process is not disruptive for the main use case, enhances visibility for the authors and possibly enlarges the number of developers/contributors. This process constitutes another possible instance of *economy of scale*.
4. Software that is deeply linked to a single experiment, like a reconstruction code for a specific detector, algorithms linked to a specific framework and such. Here also there is some room for reusability, provided that developers could be supported in their efforts by a forum where they discuss and agree on standards for the public interfaces.

INFN members manifest a particular interest in two specific areas, namely GRID software maintenance and development and simulation, even tough the arena of software used in the HEP community is certainly larger than this.

INFN has significantly contributed to the development of the GRID middleware over many years with important components such as VOMS or STORM (just to name a few) and this middleware needs still to be maintained to guarantee the requirements put forward by the current operations of the LHC experiments as well as their foreseen upgrades. Moreover, as mentioned before, some of these software components could become essential parts in future of the current Cloud stacks, representing components now missing, allowing for an integration of the Grid computing model into the Cloud paradigm. Resources for these projects could come from the Horizon2020 program, but this would require a substantial synergy between different institutions in different countries: the Software Collaboration could provide the seed for such a synergic activity.

For what concerns simulation, we consider important the following:

Multi-disciplinary software

Some of the software systems and associated projects relevant to HEP are intrinsically multi-disciplinary:

1) they require or profit from competences contributed by scientific communities other than HEP (e.g. HEP experiments would benefit from the expertise of the nuclear physics community for the simulation of hadronic interactions in calorimeters);
2) they are also used in non-HEP research, industry and applications of social interest: the variety of applications making use of software originating from HEP is beneficial to demonstrate the impact of HEP on society at large.

A notable example is Geant4, to which both characteristics of multi-disciplinarity apply.

The HEP Software Collaboration should support multi-disciplinary software with appropriate governance models and technical infrastructure. It should enable and facilitate the participation of institutes and scientists not directly associated with HEP to multi-disciplinary projects.

Risk Management

Risk analysis should be regularly performed in all areas of activity of the HEP Software Collaboration and adequate risk mitigation strategies should be adopted. Special attention should be devoted to mitigating the risk of significant conceptual, architectural, functional and technological changes to software currently in use, which could be disruptive to HEP experiments and to the advancement of science in general. A fine-grained, component-based approach is desirable to mitigate potentially disruptive risks in critical areas, such as changes in the paradigm of particle transport.

Scientific identity

The scientific research character of projects associated with the HEP Software Collaboration should be recognized. They should be considered on a peer level as research in HEP experiments, rather than services or mere support to the experiments. Scientific acknowledgement of software-oriented projects is crucial to attract distinguished scientists – rather than just technical programmers – to work at them. It is also essential to attract physics students to areas that require long training to build expertise. Recognition should also come from possibility of career advances in academia, on par with communities working in areas such as detector construction, electronics or solid state.

Evaluation

The HEP Software Collaboration itself and all its associated projects should be subject to regular peer review. Reviewers should have a documented scientific and technical background in the areas they are appointed to evaluate, and should be free from conflict of interest. The review process should be open in all its phases. Scientific achievements should be regularly documented in scholarly journals.

Finally, for what concerns duration of the Collaboration, we think it should become permanent: software never reaches a final point of development, either continues to evolve or it simply becomes obsolete. Therefore, if a viable form of collaboration could be built to harmonize the development of software tools and libraries, it would be useful to construct it as a continuous work in progress since its inception.

Governance Model

The governance model should be as thin as possible, as already stated by many, and should essentially provide a body that acts as a sort of glue or contact between developers in various fields and contexts, promoting exchange of architectural ideas or even just technical details in a harmonic way, avoiding duplication of efforts where possible and sharing of common practices and ideas in different areas.

The first priority should be to identify what benefits the collaboration could provide with respect to the existing development approach, while the identification of an appropriate model of governance could be the focus of discussion at a later stage, taking into consideration analogous approaches taken by Open Software organizations such as Apache or Boost.

Membership

Membership of the collaboration governance should be decided by each Funding Agency Management, depending on the chosen agreement, but participation in just the Technical Forum, instead, should be open to anyone interested, both as a user and as a developer. This would ensure that all voices are somehow represented, at least in exposing use-cases to the overall audience: proposals for further action could then be assembled by a thin layer of conveners, who would suggest priorities and goals to the developers, agreeing on milestones with them and leaving ample autonomy to each individual development team.