LCG Application Area Internal Review

April 2005

1. INTRODUCTION

From March 30 to April first the LCG Application Area held an internal review of the software and infrastructure areas.

The Review Committee concentrated on the design, implementation, performance and experiment experience with SPI, POOL, SEAL and Simulation software and infrastructure. It aimed to examine the overall coherence of the software, to identify real and potential problems and risks, and make recommendations on the evolution of the Application Area projects.

A particular focus of the review was on SEAL and ROOT software and the proposal to merge SEAL and ROOT into a single project. With production software now in use within the experiments, the experiments were able to provide substantial feedback to guide the direction and prioritization of future work.

The software developers of the projects made detailed presentations allowing an in-depth technical assessment of the design and implementation of the software. The experiments have given feedback on the use of the AA software and presented their opinion on the merger proposal.

Membership of the Review Committee:

Experiment representatives: Predrag Buncic, Paolo Calafiura, Marco Cattaneo, Maya Stavrianakou, William Tanenbaum, Teddy Todorov, Vakhtang Tsulaia

SC2 members: JJ Blaising, Albert De Roeck, Gerhard Raven

2. GENERAL COMMENTS

The Review Committee acknowledges the large amount of work completed in preparation for the review. The detailed and thoughtful presentations were extremely helpful for the committee.

3. SPECIFIC COMMENTS TO THE PROJECTS

3.1 SPI

The goal of the project is to provide a software infrastructure and a process for the software projects that are being deployed in the LCG Application Area.

General remarks

The impressive progress made since the last review is demonstrated by the widespread adoption of SPI supported tools and systems by several experiments and other LCG projects. Most recommendations made at the time have been implemented with visible beneficial effects especially in what concerns the functions of the central librarian.

The proposed direct participation in the projects in particular in what concerns release and quality assurance management is strongly encouraged.

The extensive suite of support tools, notably for release and distribution purposes, developed by and for SPI should be packaged and distributed for general use.
The following subsections address specific concerns and recommendations.

**Documentation**

The Doxygen/LXR documentation is generally of high quality.

*Recommendations:*

- As planned, include automated documentation building in release
- Include cross-referencing between projects
- Provide versioned documentation for external software, for instance CLHEP and AIDA, for which the API is important for the users

**Bug tracking**

The customizations brought to Savannah in collaboration with the original tool developers are commendable. The widespread use of Savannah by AA projects and experiments testifies to its strengths and further potential.

The referees are concerned about the proliferation of different systems: Savannah, Bugzilla (Geant4), ROOT bug DB (ROOT). Experiments require a coherent system, particularly for cross-referencing and migrating bugs between projects. No CERN/LCG resources should be dedicated to maintaining alternative systems.

*Recommendations:*

- Create user forum (possibly mailing list) to facilitate the exchange of ideas and experience
- Converge on means of encouraging end-user logging into Savannah
- Productize and distribute utilities for bulk submission or migration of bugs
- Productize and distribute utilities for retrieval and analysis of tracker data for reports, statistics etc
- Encourage, support and facilitate adoption by Geant4 and ROOT.

**External software**

Given the extent of adoption and use of the SPI-supported external software installations and services, several improvements should be undertaken. It must be noted that the procedure for selecting supported packages, platforms/compilers is no longer just an AF issue. Other LCG areas, especially Grid Deployment are directly concerned and affected.

*Recommendations:*

- Document procedures for the selection (and lifetime) of supported packages
- Provide “development” installations for “not-yet supported” platforms/compilers
- Provide rules/guidelines for the installation of “unsupported” packages.
- Document support commitment for above categories rather than rely on different AFS trees to distinguish
- Establish collaborative mechanisms with other LCG areas, especially Grid Deployment

**Build and distribution**

Although the choice of build tool is no longer an issue, a clear statement of strategy, which should also clarify the role of SCRAM is needed.

The needs of the experiments, for which LCG-AA projects are seen as external packages, and LCG deployment concerns can be better understood and addressed.

*Recommendations:*

- Clarify situation regarding config/make plans
- Support LCGCMT and SCRAM configuration files following tool evolution; this is especially urgent for SCRAM version 1, which has been released and adopted by CMS
• Minimize package dependencies and distinguish between intrinsic package dependencies (packages required to compile/link) and tool dependencies (packages required for e.g. testing);
• Provide different types of distributions for different needs; use LIM meeting to address requirements such as distribution granularity, build types etc
• Address LCG deployment needs
• Encourage adoption by experiments of SPI build/install/distribution scripts, tools, services; package for easy use by experiments
• Ensure representation of AA in certification discussions: Linux certification, deployment of alternative compiler etc

Quality assurance

Given the impressive suite of tools now in place in SPI, it is becoming imperative to establish QA policies and to encourage project compliance and experiment collaboration.

Recommendations:
• Establish clear QA procedures to be followed by projects
• Discuss and converge on meaningful metrics to be collected and used for quality monitoring, also as part of release procedure (e.g. a la IgNominy as used in CMS)
• Find ways to encourage compliance; possibly by deploying QA managers
• Encourage adoption by experiments, possibly by offering consultant services
• Coordinate evaluation and selection of external tools communities (e.g. valgrind vs. Intel profiling tools) in collaboration with the AA/AF
• Provide support for widely used tools (e.g. the IRST rulechecker)

Training

Recommendations:
• Training should become an SPI responsibility
• The highly successful Python course must be continued and extended with examples from recent developments (e.g. Geant4 Python interface)

3.2 POOL

The POOL project has been created to implement a common persistency framework, allowing to store and retrieve experiment data and Meta data in a distributed and grid enabled environment.

General remarks

The POOL project achieved excellent progress since the last review. Three LHC collaborations (ATLAS, CMS and LHCb) use POOL as the base for persistency. Each of these collaborations has produced large numbers of events (tens or hundreds of millions); with a total volume of data stored in POOL of about 400 TB. No major POOL related problems were encountered in the production and access to the data. POOL has successfully met the requirements of the collaborations using it. Most recommendations of the previous review have been implemented.

The merger of SEAL + ROOT will have a significant impact on POOL. This will generate an additional workload, which must be anticipated in the planning.

The following subsections address specific concerns and recommendations.

Documentation

While the POOL documentation is greatly improved since the previous review, some problems still remain. A few examples of documentation problems at the time of the review:
• Missing documentation: how to set storage technology specific options (e.g. ROOT tree split level)
• Wrong or obsolete documentation: Initialization of untyped collections is described incorrectly
• Garbled documentation: Some lists do not show up in PDF version (e.g. sections 18.4.1 and 18.6.1). These are OK in HTML version
• Incomplete documentation: Sections 2 (POOL by examples) and 4 (POOL-CORE: user level semantics) of the User Guide are empty.

Recommendations:
Drop the obsolete Workbook and concentrate on the User Guide. Add a section to the user guide on setting storage technology specific options, in particular, ROOT options, in complete detail. Give a high priority to documentation errors reported by users in Savannah.

Bugs and bug fixes
While the time to fix bugs is very good in general, there is a small number of persisting bugs that seem to be in the border area between POOL subsystems, in particular in the border area between the Storage Manager and Collections; it is not clear who will fix them.

Release process
The reviewers welcome the split of the RAL and POOL release cycles

Recommendations:
Make the release process compatible with the rest of the Applications Area

Error handling and reporting
The error messages are not sufficiently complete and informative.

Recommendations:
Improve error handling; error reporting must propagate to end user with clear indication of which component in the complicated stack encountered the error, and provide sufficient description of the error

Collections
The situation with POOL collections is confusing, mainly as a consequence of no clear requirements from the experiments. The lack of progress is not due to any failing on POOL's part. It appears that currently only CMS is using (implicit) POOL collections, and that its requirements are met.

Concern:
As experiments mature, new requirements for collections may be requested in the future. POOL should meet these requests, subject of course to manpower constraints. The experiments may need to provide manpower

Recommendations:
Try to suggest deadline for user requirements

Storage manager
The goal of full ROOT browsability for files produced by the POOL storage manager has mostly been met by ROOT 4

• Some small improvements are needed, but these are within ROOT and are already being addressed by the ROOT team.
• The ability to follow a POOL::Ref in an interactive ROOT session is needed (this work falls probably within the new ROOT/SEAL team rather than with POOL)

File catalogues
POOL API

Publishing the POOL API for the File and Metadata catalogues is an important and welcome step. The task of implementing this API should be performed by the different catalogue providers, and not necessarily by the POOL team.

Recommendations:

Keep the API stable

POOL and emerging catalogues

Which catalogue will be deployed will be ultimately decided by

- Resource/service providers (sites, grids. …)
- Virtual Organization (experiment); POOL will have to use various FC back-ends and that will result in end users experiencing different Quality of Service.

Suggestion:

Define a reference benchmark for the file catalogues to be able to assess the performances of different file catalogue implementations and to differentiate between POOL and backend problems

POOL and security

In a distributed (grid) environment, POOL should not be the weakest point in the chain

Concern: Performance with security implemented

Recommendations:

Check user requirements and solutions developed in the Grid community and other applications (PROOF)

COOL

The reviewers note favorably the evolution of the conditions database project into COOL. However COOL is very young (a first public release was imminent at the time of the review) and therefore has yet to succeed.

- There is a public commitment from 2 experiments (ATLAS and LHCb) to use COOL.
- The CMS conditions database group is considering the use of COOL, but has not made any commitment to use it. Currently, they are using POOL directly, and considering FroNTier

Suggestion:

Experiments interested in COOL are encouraged to commit more manpower to assure project survival

3.3 SEAL/ROOT

The merger between SEAL and ROOT into one ROOT-CORE project offers great opportunity to unify and consolidate the 'foundation' of the LCG-AA, providing a clear, coherent picture to (end) users -- but at the same time, during this process, there is the potential for disruption and confusion. This is obviously due to the fact that this merger concerns the 'base' or 'core' of both the LCG-AA software and ROOT, both of which are heavily used.

From the presentations shown at the Internal Review, including (but not limited to) the feedback of all four experiments, it is clear that there is a widespread consensus on the merger of the ROOT and SEAL projects, and its goals. It is clear that a coherent set of framework services, without duplication, will make it easier for (end) users. The crucial next step is to implement a detailed planning, including timescales and technical details, and, during this
process retain the consensus. Given the impact on dependent projects, this step should be taken as soon as possible.

As the four experiments are major stakeholders in the LCG-AA, and the capability to achieve their goals depends strongly on the software (infrastructure) and services provided by the AA, they should be able to define which items they consider critical, and set the priorities accordingly. These will include, but are most likely not limited to, the developments of the dictionary, math libraries, CLHEP migration and/or replacement, the plug-in manager, etc. Especially those items which will have a knock-on impact on the planning and/or requirements of dependent projects, such as POOL, should be closely monitored.

Recommendations:

To achieve this, we recommend that the prioritization and scheduling of the SEAL/ROOT project be supervised by the LCG Architects Forum (AF). In addition, we recommend that the deployment of LCG-funded persons must be in those areas deemed of highest importance by the AF. Care must be taken that non-critical items will not have a negative effect on critical ones, by, for example, requiring additional complexity.

In the process of the merger, compatibility should be maintained where possible, i.e. as long as it has no negative impact on the strategic goal(s), the planning and/or the workload.

In order for this merger to succeed, it is vital to insure that the best part of the two projects is taken forward, and it should be self-evident that both sides will have to compromise and learn from each other. This merger should not be seen as just an opportunity to 'add missing features to ROOT' and be done. Instead, the architectural strengths of SEAL should be conserved: the component model, the limited requirements on (client) objects. In this respect, lightweight packaging is crucial. It should be possible to pick certain core components (e.g. the math library, plug-in management), without having to depend on the entire framework. In addition, this will make maintaining bug-fix releases of older versions easier.

Note that this does not preclude combining the components/libraries together prior to distribution of specific packages such as the principal ROOT executable. We note that the ROOT team has already taken many steps in this direction, as is obvious from e.g. perusing the configuration file for the ROOT plug-in manager -- there are many implementations of various interfaces, which are loaded on demand, when required, at run-time by the plug-in manager. We encourage them to continue further in this direction.

It is especially important that classes representing 'basic' concepts such as (Lorentz) vectors, mathematical functions, random number generators, but also plug-in components should be usable 'stand-alone', independent and disjoint from the (ROOT) framework, as requested by three of the LCG experiments. With the release of ROOT4, it is possible to inspect, to read and write and to draw instances of so called "foreign" classes not-inheriting from ROOT TObject. We welcome this new development which should allow incorporating (SEAL) classes into the new core without introducing any explicit ROOT dependency. We also recommend that, whenever possible, the existing ROOT classes which will in turn become part of the new core libraries should be re-engineered to make them usable outside of the ROOT framework.

In case of the plug-in manager, we note that there is a substantially different approach between SEAL (which is factory based) and ROOT (which is interpreter based). One of the ingredients in the decision on how to proceed forward should be the impact on the existing experiment schemes, e.g. the Gaudi component-model. Care must be taken, as the plug-in manager scheme has a potential impact, on how the configuration of the various frameworks/applications is performed; this could have an end-user visible impact.

As concerns the evolution towards the common dictionary we observe that there is agreement on the end-product, but there are still ongoing discussions on the path towards this goal. The ordering, scheduling and priorities should be agreed upon the May workshop dedicated to the topic, and presented to the AF as soon as possible so that the work planning of dependent project, e.g. POOL can, if necessary, be adapted accordingly.
The work on the Math library integration seems most advanced. We conclude that the experiments are neutral about licensing, but are concerned about duplication as this might cause confusion on reproducibility of results. As a result, we recommend that duplication be avoided. Finally, we note that the previous remarks about packaging and dependencies apply very strongly in this case.

3.4 Simulation

Simulation is a crucial component for data analysis at the LHC. The Simulation Project in the LCG AA software area contains several activities, which have been discussed in turn.

Generator services

A huge amount of work has been done since the last review. Several new activities have emerged in the subproject, e.g. the database for generator data DBMC, and Generator Level. The generator services subproject GENSER is already actively used by ATLAS and LHCb; CMS will start to use it soon. The size of the complete GENSER package is becoming however too big to be handled efficiently. Therefore the review recommends introducing more granular packaging options for distribution. The situation with HepMC is still unclear and needs to be re-evaluated once the new release is used in production by all experiments.

Physics validation subproject

The contributions from the experiments to the validation subproject have been significantly increased during the past 18 months. Geant4 physics list have been compared in detail to test beam data, and FLUKA has also been used for a number of comparisons.

Recommendations:

- A continuing common effort together with the experiments to validate the packages against test beam data and simple benchmark processes.
- Continuation of the Fluka validation, especially of the hadronic physics
- To install a central repository for the validation results
- To revisit together with the experiments the effect on systematic uncertainties for future measurements, due to the present simulation uncertainties, as observed by the ongoing validation studies.

Concern:

The reduction in manpower of 2.3 to 0.8 FTE in the physics validation of Geant4 is very worrying. If Geant4 can argue that this is essentially a CERN responsibility we recommend that the LGC tries to improve the manpower situation.

Geant4

Geant4 has proven that it has reached a level of maturity in terms of performance, physics results, reliability and stability. The communication and interaction with the LHC experiments is also noted to be very good. Geant4 has become the main simulation engine for the experiments ATLAS, CMS and LHCb. All have expressed their satisfaction with the present state of the subproject.

Any concerns on the validity of some aspects of the physics results, should be brought forward in the validation subproject and examined there.

Potential issues connected with different strategies on the usage of CLHEP libraries by Geant4 and other LCG applications should be addressed, and may require immediate solutions.
FLUKA

Alice plans to use FLUKA as its main simulation engine. The other LHC experiments confirm their interest to use it in future as a second engine or for special studies. FLUKA should be usable via a fully functional, available and documented conversion mechanism (FLUGG as an already working example)

Recommendations:

- FLUKA installation together with source code should be made available very soon
- Distribution of FLUKA should make use of the SPI tools, like other AA related software products

Generic simulation framework project

The simulation framework packages have so far been mainly used internally within the simulation project.

- At this time there is no interest from the experiments to have a common generic simulation framework and new developments are therefore not encouraged.
- VMC remains a viable solution in the case such interest will be expressed by more than one experiment

Further development of GDML is encouraged (missing elements, modularization etc...) Some experiments have already expressed their interest in GDML; its usage is also included in the Geant4 planning as a geometry interchange format and possible solution for persistency.

The Python interface to Geant4 needs some effort in creating the documentation. It is recommended to exchange experience in this field with experiments that already have their own and well advanced solutions, e.g. ATLAS.

The MCTruth is considered to be an important part of the project and a LHC experiment-wide common development in this area is strongly encouraged.

There is however some concern that this whole subproject is carried by less than one FTE. In the long run this may be insufficient.

3.5 Miscellaneous

The experiments are developing their distributed data analysis frameworks and get support from ARDA. POOL and PROOF are AA projects and clients of grid services.

The review noted the absence of a technical forum to discuss client/services issues between the experiments, the AA developers and the grid service providers.

Recommendation:

Organize regular extended Architect’s forum meetings to establish better communication and working relationship between the relevant parties.

CONCLUSION

The Review Committee was impressed by the work that has been done since the last review. Most recommendations made at the time have been implemented or are in the plan.

The proposed evolution plan is technically reasonable and supported by all experiments. It should allow integrating ROOT activity in LCG organization, providing a coherent set of products to the users, consolidating the projects and facilitating the long term support.

The technical details of the plan should continue to be discussed and approved by the Architects Forum. LCG-SC2 should follow up closely the progress of the SEAL/ROOT merger.