Comments to SPI

General remarks

- Impressed by progress since last review
  - Widespread adoption by experiments and projects
    - Savannah, ExtSoft
  - Build system no longer an issue
- Most recommendations implemented
  - Benefits of central librarian already visible
    - LIM meeting has useful role
- Welcome direct participation in projects
  - Release manager, QA manager
- Recurring observation:
  - Tools developed for SPI use should be packaged for general use
    - In particular in release and distribution areas

Doxygen and Savannah

- Doxygen/LXR documentation
  - Currently produced manually, should be automated as part of release procedure (planned).
  - Cross-referencing between projects (produce and use doxygen tagfiles)
  - Document also external libraries where API is important to users (e.g. CLHEP, AIDA)
- Savannah
  - User forum (mailing list) would be useful
    - Self help (a la root-talk)
    - Exchange of ideas and experience – e.g. how to encourage logging into Savannah by end users of software
  - Strong interest in utilities for bulk submission or migration and retrieval/analysis of tracker data for reports, statistics etc
  - Concerned about the proliferation of different systems
    - Savannah, Bugzilla (G4), Root bug DB (ROOT+SEAL?)
    - 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 (e.g. Bugzilla for G4)

External software

- Document procedures for:
  - Selection (and lifetime) of supported packages
  - Providing "development" installations for "not-yet supported" platforms/compilers
  - Installation of "unsupported" packages
    - Can we avoid sw/contrib?
- Document support commitment for above categories
  - Do not use different AFS tree to distinguish
- Procedure for selecting supported packages, platforms/compilers is no longer just AF issue
  - Find mechanisms to include other LCG areas, especially Grid Deployment

Build and distribution

- Choice of build tool is no longer an issue
  - But need clear statement of strategy
    - What is role of SCRAM? What happened to config/make plans?
    - LCGCMT and SCRAM configuration files must follow tool evolution.
- Minimize package dependencies
  - Make distinction between
    - intrinsic package dependencies (packages required to compile/link)
    - tool dependencies (packages required for e.g. testing)
- Provide different types distributions for different needs
  - Use LIM meeting to define needs
    - Granularity of distribution, build from sources etc.
    - Expts. see LCG-AA projects as external packages
  - Address LCG deployment needs
- Encourage adoption by experiments of SPI build/install/distribution scripts, tools, services
  - Package for easy use by experiments
    - Use LIM meeting to address requirements
- Representation of AA in certification discussions
  - Linux certification, deployment of alternative compilers

QA

- Impressive suite of tools now in place
  - 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 (QA managers)
  - Encourage adoption by experiments
    - Consultant role?
- Coordination role for selection of external tools
  - Evaluate new tools, recommend to AA/AF community
    - E.g. valgrind vs. Intel profiling tools
  - Support for widely used tools
    - E.g. CodeWizard vs. RuleCheck

Training

- Training is not currently a responsibility of SPI
  - It should be
- Highly successful Python course must be continued
  - And extended with examples from recent developments (e.g. Geant4 Python interface)