SPI
Software Process & Infrastructure

Project Status

http://spi.cern.ch

Application Area Review – 30 March 2005

Presentation Outline

2003

- Recommendations of previous Review

2004

- Status of the SPI Services
- Actions performed to fulfill the recommendations
  - **Savannah**
  - **Ext. Software**
  - **Build and Release**
  - **Testing and QA**
- Milestones 2004

2005

- Summary and focus 2005

Recommendations of 2003 review (I)

- **CVS service**
  - Move to the IT CVS service as soon as possible
  - After testing of features and performance
  - Use centrally managed tools
- **Savannah portal**
  - Avoid divergence of CERN customization with mainstream Savannah
  - Concerns about scalability
- **Documentation and web site**
  - Website and SPI workbook were praised as good and complete
  - Maintain workbook up to date
  - Verify that doxygen comments are included in the code

Recommendations of 2003 review (II)

- **External Software service**
  - Transparent decisions making for provision and maintenance of external software
  - Add flexibility to support different structures of installations and maintain the original structures of the tools
  - Make compilation and installation scripts available
  - Suggest simple QA scripts to validate installations of new versions of external tools
- **QA activities**
  - Work to automate the verification
  - Person to be centrally responsible for QA and to chase projects to improve compliance

Recommendations of 2003 review (III)

- **Build System and Infrastructure**
  - Role of a central librarian
  - Common build settings, release procedures
  - Share of common build tools not developed within the projects
  - Build the software centrally and free developers in projects
  - Have prereleases available to the users
  - Investigate a config/make solution
  - Clarify long term strategy
  - Support others build system used by other experiments (CMT)
- **Software Distribution**
  - More interaction with LCG deployment and check existing distribution tools (e.g. pacman)
  - Customize distribution by platform and component

SPI Services

- External Software
- Savannah Project Portal
- Software Librarian, builds and releases
- Testing Frameworks
- QA checklists and reports
- Software Distribution
- LCG Software Configuration
- Development of LCG policies, templates
- Code Documentation (doxygen, lxr)
- Documentation and LCG Workbook

Actions to fulfill the recommendations

- **CVS service to IT**
  - Passed to IT after validation and intensive testing
  - Migrated all projects ourselves, no problems
  - Provided users with scripts to fully support the transition phase

- **Documentation and web site**
  - Doxygen and LXR are run regularly and up to date.

  - Our website is stable and we keep it up to date
  - Using wiki and other systems to simplify writing of documentation from several platforms
  - No documentation standards achieved in the field but improvements

  - Trying a few solutions (DocBook, Wiki, blog)
  - Maybe too early in these heavy development phases

Main SPI Services

- Savannah Project Portal (Y.Perrin)
- External Software and Configuration (E.Poinsignon)
- Build and Release (A.Pfeiffer)
- Testing and QA (J.Benard)

- Avoid to develop our own tools, use available open source tools (savannah, qmtest, cppunit, lcov, pacman, etc.)
- General solutions but use them immediately for some projects that needed them
- Automate whenever it can be done
- User support and maintenance are a constant factor
- We always try to make the users happy and don't follow blindly the rules (several special builds, external packages, options, etc.)

→ Savannah Project Portal

- Follow-ups of the 2003 Review
- User support
- Trackers enhancements
- Other consolidation
- Status
- Next improvements

Follow-ups of the 2003 review

- **Recommendations of the 2003 review**
  - Avoid divergence of CERN customization with mainstream Savannah
  - Concerns about scalability
- **Resulting actions**
  - Close collaboration with the main Savane ("re-brand" of the open source) developer
  - LCG developments done on separate branch and regular re-synch with the open source
    - **February 2004**: '2003' CERN and open sources merged
    - **March -Nov 2004**: new developments triggered by
      - LCG and EGEE requirements
      - pending support requests
    - **November 2004**: '2004' CERN and open sources merged
  - Scalability: users: 399 → **1083**, projects: 68 → **122**

  … no threat in view

User support

Number of items opened & closed since last review:

- **36 bugs fixed** (main ones listed later on)
- **29 support requests handled**
  - 18 requests for assistance
  - 6 requests for new features (JRA1 middleware, EGEE/PTF, Grid Deployment, etc)
  - 5 invalid postings
- **41 tasks**
  - 22 consolidation (new functionality & major fixes)
  - 7 LCG & Open Source merge
  - 12 project approval and initialization
- **Constant direct user support**
  - Few per week

Trackers enhancements (1)

- common tracker for bugs, support and tasks
  - same level of functionality, parameterization, notification
  - common user interface
  - cross tracker and cross project links and dependencies
  - item migration between trackers & between projects
- concept of 'private items' introduced
- enhanced control of submission rights (e.g. logged-in users)
- configurability of display order (most recent <-->oldest)
- enhanced notifications
  - more granularity (category related, state changes)
  - new types of notifications (e.g. pending items reminders, news)
  - configurability of subject line
  - optional addition of project, tracker, item id in email headers

Trackers enhancements (2)

- **implementation of a 'state transitions' manager to control:**
  - any field with predefined values (select box)
  - in any tracker
  - applies to any value transition and initial value assignment
  - project administrators specify:
    - allowed/forbidden transitions
    - automatic update of other fields (e.g.re-assignment of responsibility / closure of item)
    - related notifications

Trackers enhancements (3)

Field Label: Status [Jump to this field usage]

Existing Values

Value Label	Description	Rank	Status	Occurences
	---- ACTIVE VALUES ----			
In progress	The bug is being worked on	70	Active	35
Unreproducible	The project team was unable to reproduce the bug	70	Active	1
Duplicate	This bug is already covered by another bug description (see related bugs list)	80	Active	39
Ready for Integration	A bug fix is available and can be added to next build	80	Active	93
Ready for Test	A bug fix is included in the current build and ready to be tested	90	Active	305
Ready for Review	The bug fix is ready for review and approval by the originator	150	Active	5

Registered Transitions

From	To	Is Allowed	Others Fields Update	Carbon-Copy List	Delete?
None	Wont Fix	Yes	Open/Closed:Closed		
Ready for Test	Ready for Review	Yes	Edit others fields update		
In progress	Ready for Integration	Yes	Assigned to:iteam		
Ready for Integration	Ready for Test	Yes	Assigned to:egeetest		
Accepted	In progress	Yes	Edit others fields update		
None	Accepted	Yes	Edit others fields update		

Other improvements

- **item search by ID number**
- **project members back-ups** (notifications)
- **duplication of project configuration**
- **login /password consolidated**
- **multiple submissions of same item fixed** (introduction of form ids)
- **data no longer lost after submission of incomplete form**
  (now returns user filled form instead of error page)

- **item submission by program** (not integrated in savannah)
  - useful for bulk submission / migration to savannah from other systems
  - configurable (server, project, tracker) perl script
  - uses the www-mechanize API.
- **retrieval of tracker data to perform reports, statistics, etc** (not integrated in savannah)
  - ad-hoc implementation for the LCG2sites needs
  - general savannah export facility is needed

Status

LCG Savannah: Statistics

Total: 1083 users 122 projects

Since 2003-Oct-20 00:00 until 2005-Mar-23 00:00

Accounts: 684 new users 54 new projects

New / Total:

users	684/1083	63%
groups	54/122	44%

Trackers: 8013 new item(s), including 4882 already closed

- 614 new support request(s), including 104 already closed
- 5789 new bug(s), including 3612 already closed
- 1357 new task(s), including 966 already closed
- 253 new patch(es), including 200 already closed

New items per Tracker / Tracker Total:

Support Request(s)	614/767	80%
Bug(s)	5789/7478	77%
Task(s)	1357/1749	77%
Patch(es)	253/260	97%

Next improvements

- **Documentation**
  - needs to be improved
  - many features are unknown by the users
- **Functionality**
  - general ‘export’ facility (possibly based on XML) to be studied and implemented
- **Communication with the user community**
  - a ‘light’ mechanism to interact with such a large community is needed (FAQ, Wiki, Blogs, etc)

→ External Software

- **Recommendations of 2003 review**
- **Improvements in:**
  - External Software
  - Dependencies
  - Automatic installation of external
  - Automated web
- **Foreseen improvements**

Recommendation and action

- **Recommendation on the External Software service**
  - Transparent decisions making for provision and maintenance of external software
  - Add flexibility to support different structures of installations and maintain the original structures of the tools
  - Make available compilation and installation scripts
  - Suggest simple QA scripts to validate installations of new versions of external tools

Action performed to fulfill the recommendation
External Software service

- Maintain original structure of the installed packages
- Automated installations and making available all installation scripts and tools
- Some tools for which we have verification material we now check the installation, more on the way
- Major improvements

External Software

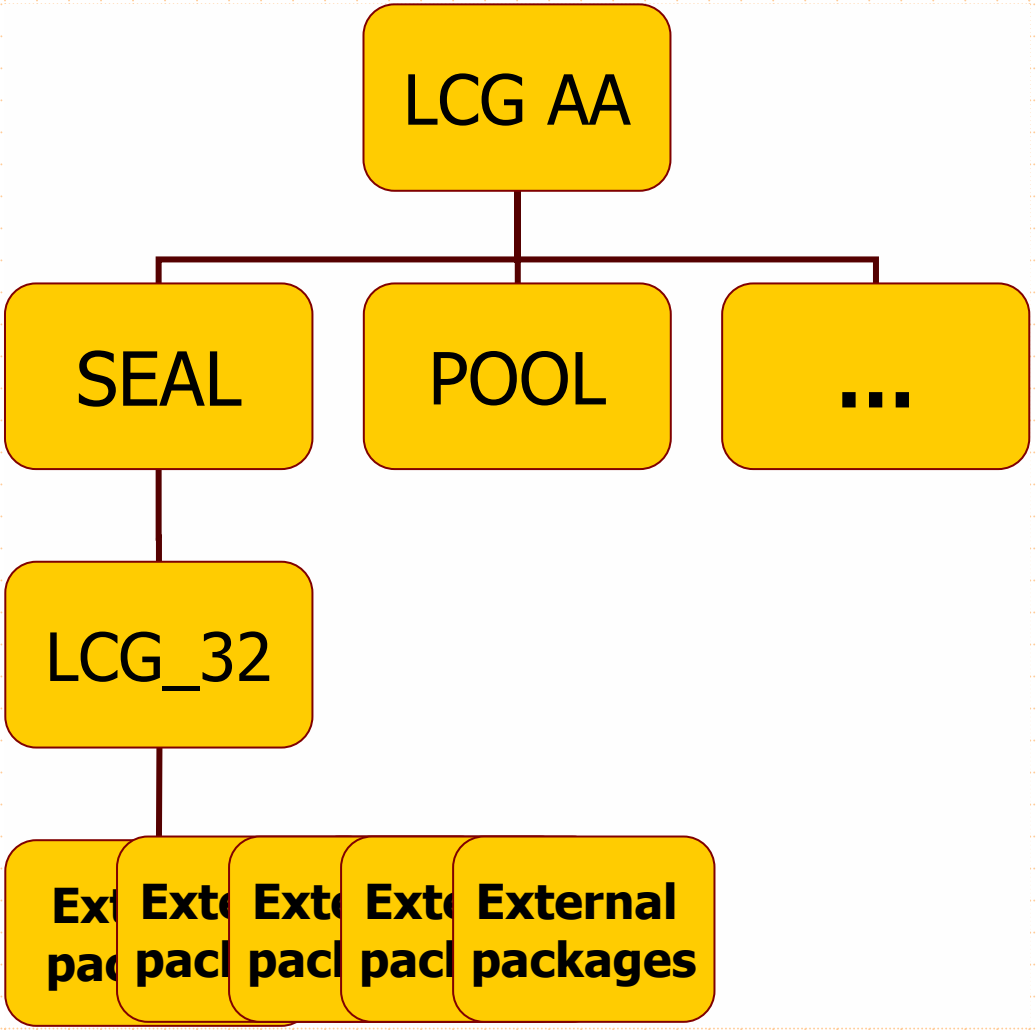
- ~80 different packages following the **same structure**:
  /afs/cern.ch/sw/lcg/external/<package>/<version>/<platform>/
- New platforms:
  - **slc3_ia32_gcc323**
    (certifying the temporary cel3-i386_gcc323)
  - **osx103_gcc33**
  - Still providing:
    - rh73_gcc32(3),
    - rh73_icc80,
    - win32_vc71
- Total of 500 installations on 100GB.
- The **planning** of the next installations is public and accessible from the News page

External Software planning

External software	Dep	Version	rh73_gcc32	rh73_gcc323	rh73_icc80	rh73_ecc80	cel3-slc3_ia32_gcc323	win32_vc71	osx103_gcc33	platform independ
pcre	S P Pi	4.4		Done	Done		Done		Done	
		4.5		Done						
psyco		1.2	Done	Done			Done		not mac	
pyqt										
Python	S P Pi	2.2.2		Done	Done		Done			
		2.3.3	Done	Done	Done		Done	Done	Done	
		2.3.4	Done	Done	Done		Done	Done	Done	
python_packages		23_LCG26	X	Done					Done	
QMtest	S P	2.0.3		Done			Done		Done	
		2.2.1		Done	S		Done	Done	Done	
qt		3.3.2		Done	Done		Done		Done (mac)	
qutexmlrpc		0.2_qt332	Done	Done	Done		Done		Done	
root	S P Pi	3.10.02		Done	S		Done		Done (no dcap)	
		4.00.03	Done	Done	Done					
		4.00.08	S	Done	S		Done	Done	Done (no dcap)	
		4.00.08b	Done	Done	S		Done	Done	Done (no dcap)	
		4.01.02	Done	Done	Done		Done	Done	Done (no dcap)	
		4.03.02	Done	Done	Done		Done	Done	Done (no dcap)	
rulechecker										
sherpa		1.0.5	Done	Done			Done			

Dependencies

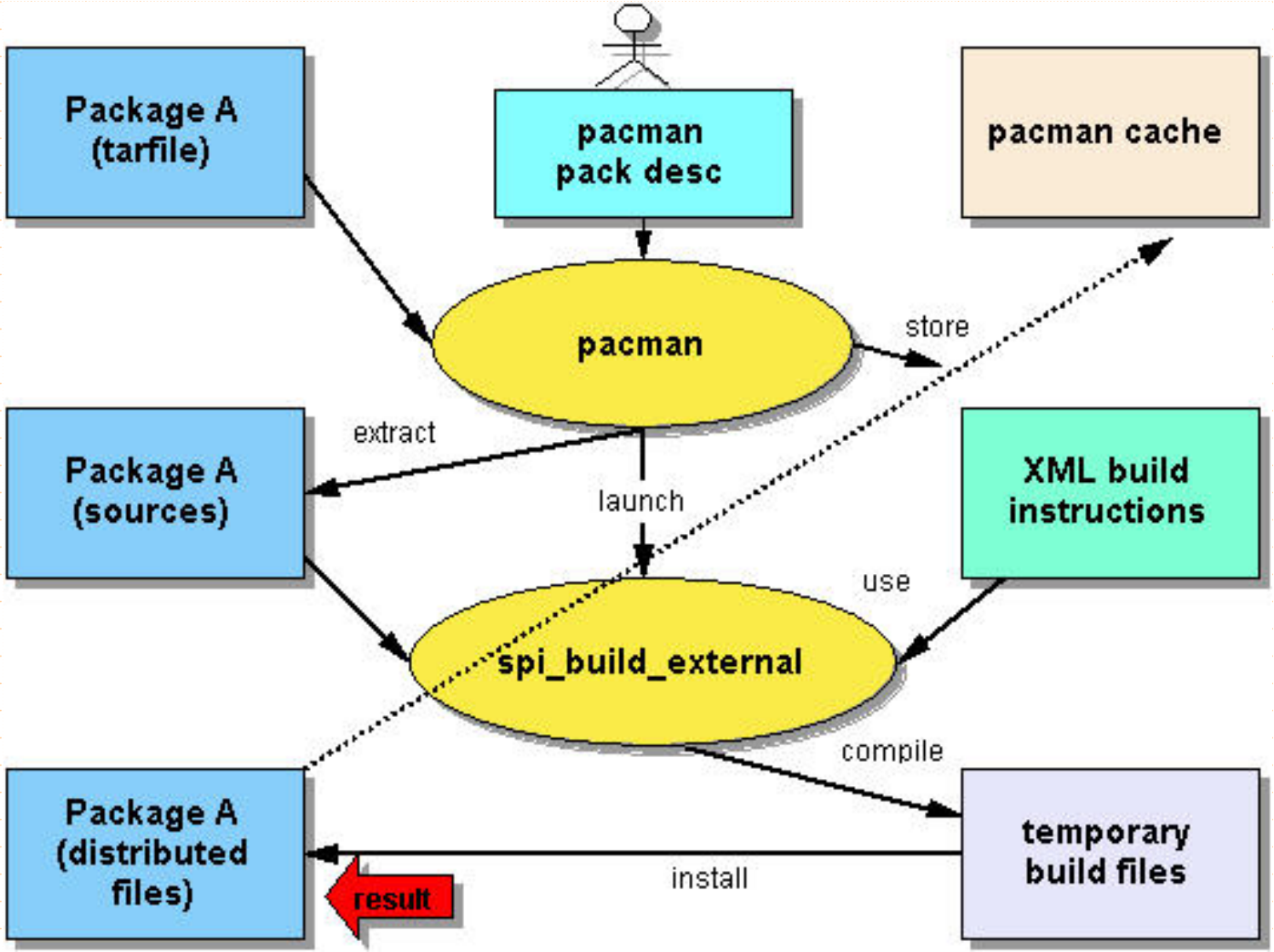
- **XML** files describing the **dependency** tree from the Application Area projects to the external packages.
- **Simple** representation, **independent** of specific formats
- One XML file per link which will be used to generate all that is needed: web, distribution, CM, ...



Dependencies, XML example

```
<package name="external" version="lcg32" tag="LCG_32">
<dependency name="gcc3" version="3.2" platform="rh73_gcc32"/>
<dependency name="gcc3" version="3.2.3" platform="rh73_gcc323"/>
<dependency name="gcc3" version="3.2.3" platform="cel3-i386_gcc323"/>
<dependency name="osxgcc" version="3.3" platform="osx103_gcc33"/>
<!-- seal 1.4.3 -->
<dependency name="uuid" version="1.32"/>
<dependency name="gccxml" version="0.6.0_patch1"/>
<dependency name="boost" version="1.31.0"/>
<dependency name="clhep" version="1.9.1.2_spi1"/>
<dependency name="gsl" version="1.5"/>
<dependency name="python" version="2.3.4"/>
<dependency name="root" version="4.03.02"/>
<dependency name="cppunit" version="1.8.0"/>
<dependency name="oval" version="3.5.0"/>
<dependency name="valgrind" version="2.0.0"/>
<dependency name="qmtest" version="2.2.1"/>
<dependency name="zlib" version="1.1.4"/>
<dependency name="bz2lib" version="1.0.2"/>
<dependency name="pcre" version="4.4"/>
<dependency name="sockets" version="1.0"/>
```

Automatic installation of External



Automatic installation of External (2)

- Interface is **Pacman**
- Pacman prepares the environment and call SBE
- **SPI Build External** (SBE) is the machinery
- SBE runs **each step of the build** following instructions stored in **xml** format
- At the end, Pacman stores information in its **cache**.
- Can be used to make installation **elsewhere**

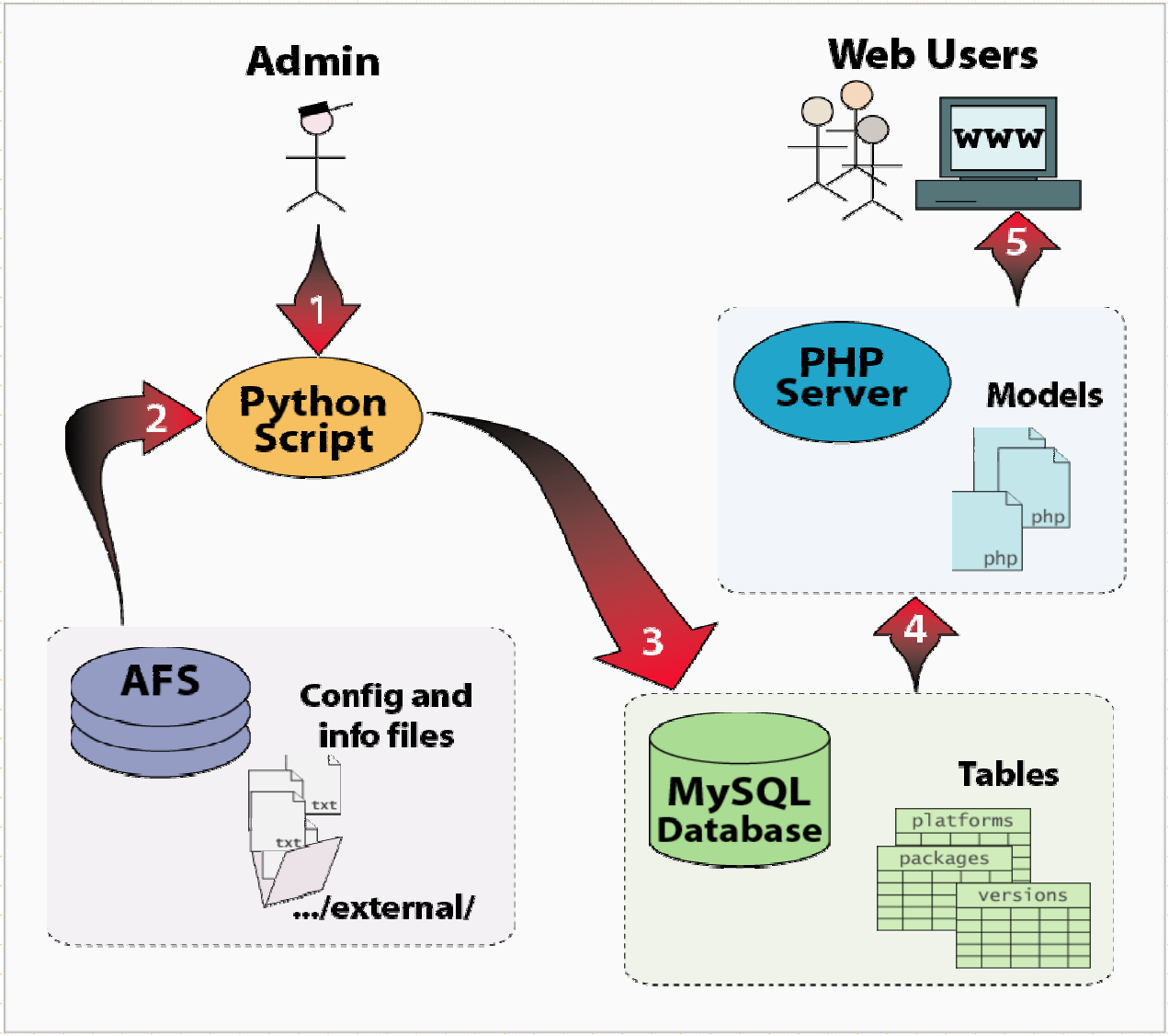
Pacman package description example

```
# more bz2lib-1_0_2.pacman

description = 'Package bz2lib version 1.0.2.'
url = 'ftp://sources.redhat.com/pub/bzip2/v102/bzip2-1.0.2.tar.gz'

setenvTemp('SPI_PACKAGE','bz2lib')
setenvTemp('SPI_VERSION','1.0.2')

mkdir("${SPI_PACKAGE}-${SPI_VERSION}")
cd("${SPI_PACKAGE}-${SPI_VERSION}")

download = {'*' : 'bzip2-1.0.2.tar.gz'}

install   = { '*' :       ['spi_build_external.py --platform=${SPI_PLATFORM} --
    package=${SPI_PACKAGE} --version=${SPI_VERSION} --
    topdir=${SPI_TOPDIR}'] }
uninstall =       ['spi_build_external.py --remove --platform=${SPI_PLATFORM} --
    package=${SPI_PACKAGE} --version=${SPI_VERSION} --
    topdir=${SPI_TOPDIR}']

cd()
```

Automated web

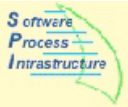
- Former web site : ~100 **html** files.
- Now : 10 **PHP** files.
- All information is up to date on AFS
- Information of all packages are stored in a **MySQL** database to able different views
- Useful improvements inside the pages like:
  - Direct access to **build info**
  - **Dependencies** table for each package
  - Customization options
- Same URL: http://spi.cern.ch/extsoft/

Automated web (2)



External Software HomePage

LHC Computing Grid > Application Area > Software Process & Infrastructure



External Software Service

Home | News | How to | Contact us | Search

LCG Software

Download Area

External Software

Alphabetic order
Platforms table
Used in LCG Projects

SPI Quick Links

SPI Home
SPI Index

Projects Portal

LCG App. Area

Home Page
LCG Agenda

PI Project
POOL Project
Simulation Project
SEAL Project
SPI Project

External Links

CERN
EP Division
IT Division
LCG

The purpose of the **External Software Service** is to provide software tools and libraries to LCG development teams.

External Software alphabetic list.

78 packages found in the database.

AIDA	Abstract Interfaces for Data Analysis.
Anaphe	C++ libraries for HEP computing.
Ant	Apache Ant is a Java-based build tool. In theory, it is kind of like Make, but without Make's wrinkles.
autoconf	Package that produce shell scripts to automatically configure software source
blas	Basic Linear Algebra Subprograms
Boost	Portable C++ source libraries.
bz2lib	High-quality data compressor library.
callgrind	Profiler plug-in of Valgrind
cernlib	CERN Program Library
clhep	A Class Library for High Energy Physics.
coin3d	Set of libraries used for creating 3D graphics applications
Colt	Open Source Libraries for High Performance Scientific and Technical Computing in Java.
comphep	HEP evaluation package
CppUnit	The C++ Unit Test Library
dcap	System for storing and retrieving huge amounts of data

External Software, one package page

LHC Computing Grid > Application Area > Software Process & Infrastructure

External Software Service

Home | News | How to | Contact us | Search

LCG Software

Download Area

External Software

Alphabetic order
Platforms table
Used in LCG Projects

SPI Quick Links

SPI Home
SPI Index

Projects Portal

LCG App. Area

Home Page
LCG Agenda

PI Project
POOL Project
Simulation Project
SEAL Project
SPI Project

External Links

CERN
EP Division
IT Division
LCG

LHC experiments

bz2lib

High-quality data compressor library.

Description

bzip2 is a **high-quality data compressor**. It typically compresses files to within 10% to 15% of the best available techniques (the PPM family of statistical compressors), whilst being around twice as fast at compression and six times faster at decompression.

Availability

/afs/cern.ch/sw/lcg/external/bz2lib/1.0.2/rh73_gcc32/ info

- install_log.txt
- start.csh
- install.txt
- start.sh

/afs/cern.ch/sw/lcg/external/bz2lib/1.0.2/rh73_ecc71/ info
/afs/cern.ch/sw/lcg/external/bz2lib/1.0.2/win32_vc7/ info
/afs/cern.ch/sw/lcg/external/bz2lib/1.0.2/rh73_icc71/ info
/afs/cern.ch/sw/lcg/external/bz2lib/1.0.2/rh73_gcc323/ info
/afs/cern.ch/sw/lcg/external/bz2lib/1.0.2/cel3-i386_gcc323/ info
/afs/cern.ch/sw/lcg/external/bz2lib/1.0.2/rh73_icc80/ info
/afs/cern.ch/sw/lcg/external/bz2lib/1.0.2/osx103_gcc33/ info
/afs/cern.ch/sw/lcg/external/bz2lib/1.0.2/slc3_ia32_gcc323/ info

Usages and Dependencies

Only the 3 last iterations are shown.

Version	Used by	Depends on
bz2lib [1.0.2]	LCG_32 LCG_31b LCG_31	

External Software, platform table

LHC Computing Grid > Application Area > Software Process & Infrastructure

External Software Service

Home | News | How to | Contact us | Search

LCG Software

Download Area

External Software

Alphabetic order
Platforms table
Used in LCG Projects

SPI Quick Links

SPI Home
SPI Index

Projects Portal

LCG App. Area

Home Page
LCG Agenda

PI Project
POOL Project
Simulation Project
SEAL Project

The purpose of the **External Software Service** is to provide software tools and libraries to LCG development teams.

External Software platform list.

58 packages found in the database.

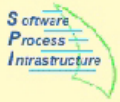
This table is limited to the last 3 versions.

Customize view or choose the default/full table of packages.

External software	Linux		Windows	Mac OS X	Platform Independent
	rh73_gcc323	slc3_ia32_gcc323	win32_vc71	osx103_gcc33	
AIDA		3.2.1			3.2.1 3.0.0
Ant					1.5.1
autoconf	2.59	2.59		2.59	
Boost	1.31.0_python233 1.31.0 1.30.2	1.31.0	1.31.0_python233 1.31.0	spitest 1.31.0_python233 1.31.0	
bz2lib	1.0.2	1.0.2		1.0.2	
callgrind	0.9.10	0.9.10			
cernlib	2004	2004		2003	
clhep	2.0.1.1 2.0.0.2 1.9.1.2_spi1	1.9.1.2_spi1 1.9.1.2 1.9.1.1	2.0.0.2 1.9.1.2_spi1 1.9.1.2	2.0.1.1 2.0.0.2 1.9.1.2_spi1	
coin3d	2.3.0	2.3.0			
Colt					1.0.2
CppUnit	1.8.0	1.8.0	1.8.0	1.8.0	
doxygen	1.4.1	1.4.1			

External Software, platform options

LHC Computing Grid > Application Area > Software Process & Infrastructure



External Software Service

Home | News | How to | Contact us | Search

LCG Software

Download Area

External Software

Alphabetic order
Platforms table
Used in LCG Projects

SPI Quick Links

SPI Home
SPI Index

Projects Portal

LCG App. Area

Home Page
LCG Agenda

PI Project
POOL Project
Simulation Project
SEAL Project
SPI Project

External Links

CERN
EP Division
IT Division
LCG

LHC experiments

Customize the External Software platform list.

Default table of packages.

Full table of packages.

Show last 3 versions only.

Print header row every 15 lines.

☑ Linux	☑ Windows	☑ Mac OS X	☑ Platform Independent
☐ cel3-i386_gcc323	☐ cygwin15_vc71	☑ osx103_gcc33	shared
☐ Linux+2.4	☐ win32_bc6		
☐ rh61_gcc2952	☐ win32_vc5		
☐ rh72_gcc2952	☐ win32_vc6		
☐ rh73	☐ win32_vc7		
☐ rh73_ecc71	☑ win32_vc71		
☐ rh73_ecc80			
☐ rh73_gcc2952			
☐ rh73_gcc32			
☑ rh73_gcc323			
☐ rh73_ia32			
☐ rh73_ia64			
☐ rh73_icc71			
☐ rh73_icc80			
☐ rh90_gcc322			
☐ slc3-i386_gcc323			
☐ slc3_amd64_gcc323			
☐ slc3_gcc323			
☑ slc3_ia32_gcc323			
☐ slc3_ia64_gcc323			

Future/Specific/Synonym
Obsolete

☐ Show all packages (Even if not available for the platforms you choose).

Foreseen improvements

- Reorganize the **Distribution** process:
  - **smaller tar files,**
  - **bigger history,**
  - **deal with different use cases like development and deployment**
- Automate the generation of External software **news**
- Automate the generation of the LCG download **web**
- Check that all processes are working properly with **AFS Replicas**
- Include the **doxygen/lxr** generation in the distribution process
- Move to **SCRAM v1**

→ Build and Distribution

Recommendations of 2003 review

- **Build System and Infrastructure**
  - Role of a central librarian
  - Common build settings, release procedures
  - Share of common build tools not developed within the projects
  - Build centrally the software and free developers in projects
  - Have prereleases available to the users
  - Investigate a config/make solution
  - Clarify long term strategy
  - Support others build system used by other experiments (CMT)
- **Software Distribution**
  - More interaction with LCG deployment and check existing distribution tools (e.g. pacman)
  - Customize distribution by platform and component

LCG Librarian

- **Started in Summer 2004**
- **Automatize the various tasks needed to build LCG software**
  - Internal (project) and external (packages)
- **Task to build the LCG software for all supported platforms**
  - Developed simple tools to automate the build of the LCG software
  - Scripts available through standard SPI installation
- **Coordinate releases and pre-releases with the projects and experiments**
  - Librarians and Integrators Meeting (LIM)
- **Maintain the configuration for the build and for other build systems**
  - E.g. generate and keep up to date the CMT files of the configurations

Actions performed to fulfill the recommendation

Build System and Infrastructure

- Librarian role filled in June 2004 and very beneficial to centralize this activity
- Developed scripts to centrally do the releases and installations
- We performed the certification of all external and LCG software for SLC3, as test of the automatic installation scripts

Librarians and Integrators Meetings (LIM)

- **Created in October 2004**
  - Mandate: Coordinate releases and pre-releases with the projects and experiments
- **Meeting every two weeks with experiments representatives**
  - Discuss/agrees the additional needs of the experiments in terms of build and package versions/installations.
- **Reports to Architects Forum that defines the general priorities for SPI**
  - AF to endorse new packages/versions required by experiments through LIM

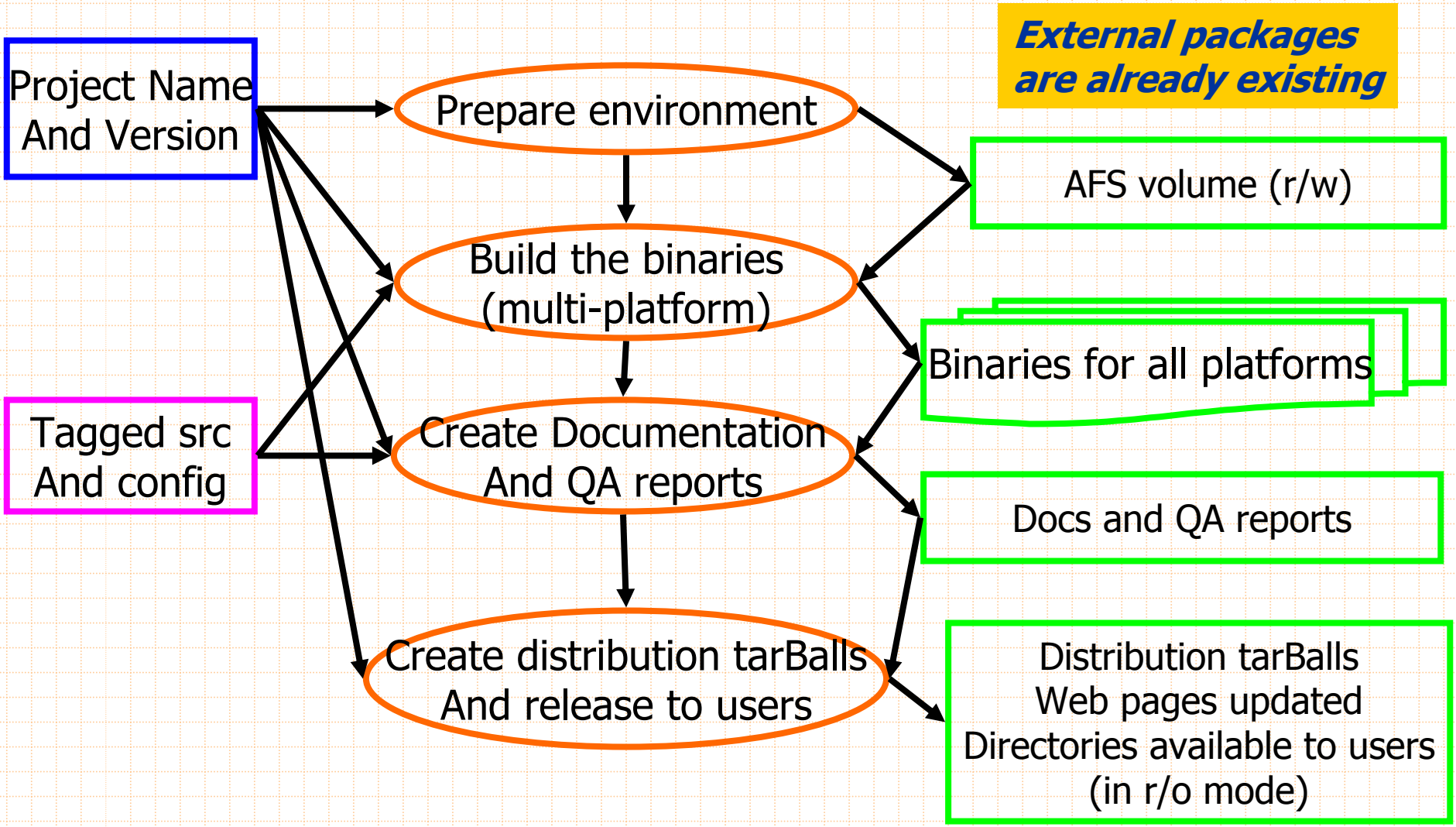
Build Servers

- **SPI maintains a list of build servers (hosted in IT) for all the platforms used in LCG AA**
  - Supported: slc3_ia32_gcc323, rh73_gcc32, rh73_gcc323, win32_vc71
  - Future/explored: osx103_gcc33, slc3_ia32_gcc343, slc3_amd64_gcc343, slc3_ia64_gcc343
  - Web page in SPI wiki
  - Linked from the SPI pages, “Build Servers”

Certification and Porting

- **Certification of SLC3 in summer/fall 2004**
- **Rebuild all packages for slc3 from scratch in December**
  - On a clean, certified platform
  - Requested by the experiments
- **Porting to new platforms**
  - Gcc 3.4.3
  - AMD64, IA64 (with gcc 3.4.3)
  - Mac OS X 10.4 (Tiger)
- **Profit from automated procedures and scripts to build everything**
  - Some cost to create the scripts and prepare the environment
    - Dependency order
  - Large gain in efficiency doing the work
  - Building all external packages from scratch takes several hours
  - “Re-use” of knowledge :-)

Building AA projects at CERN



SPI Software Distribution Service

- **Complex use cases and requirements from (mainly outside) users**
  - ***Binary*** distributions for supported (or compatible) platforms
  - ***Source*** distributions
    - Re-build on supported platform
    - Build on non-supported (but needed) platform
  - Distributions for ***developers***
    - All the s/w of a given project + required dependent packages
  - Distributions for ***batch*** farms
    - Only the “parts” which are needed for batch
  - Distributions for “***remote central installation***”
    - All the s/w for all projects + dependend packages

SPI Software Distribution Service

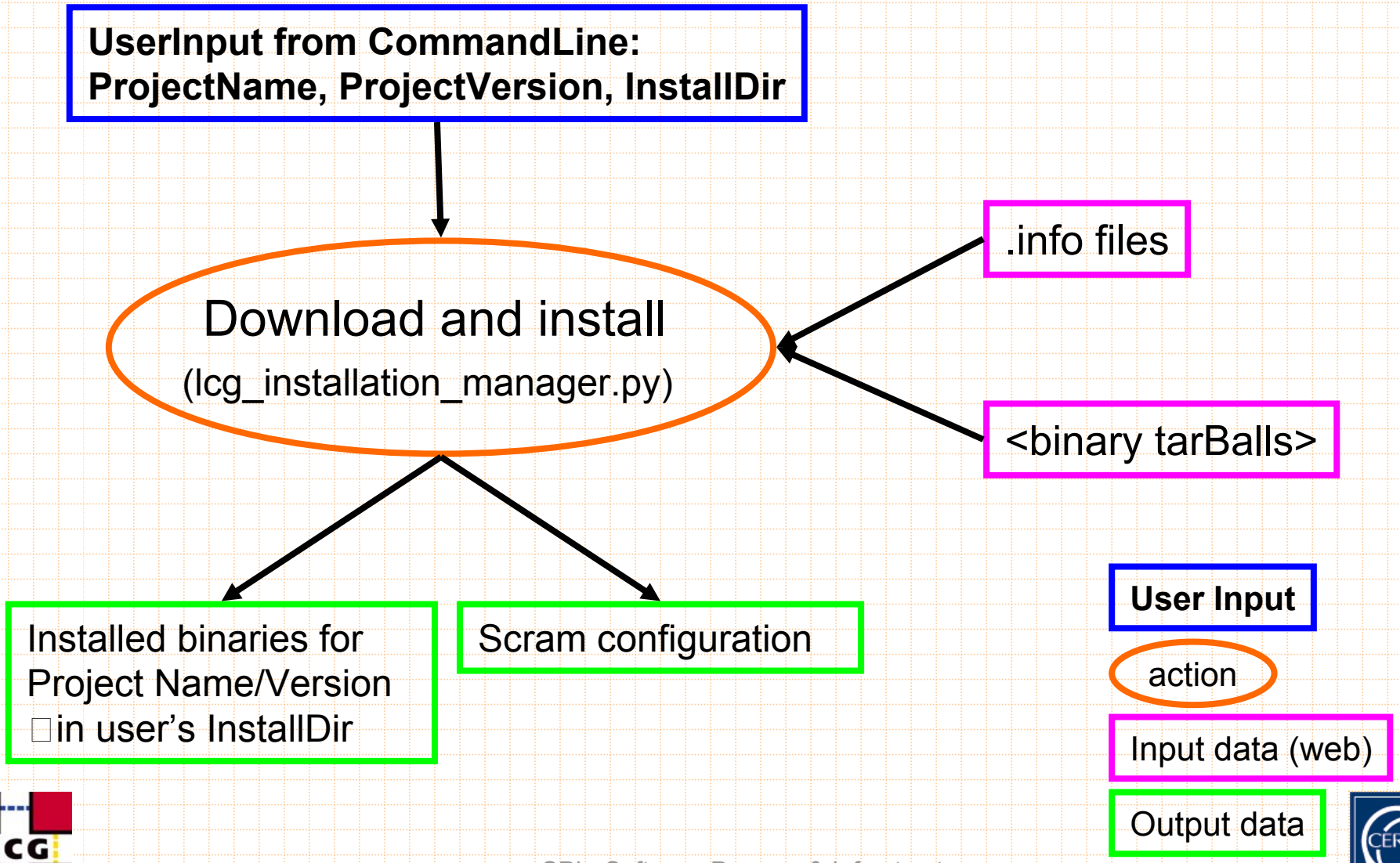
- **Inside users using AFS area**
  - /afs/cern.ch/sw/lcg/external/ and /afs/cern.ch/sw/lcg/app/releases/
- **Problem of supporting many changing needs and competing tools**
  - Complex dependencies and configuration information
  - Choose one as reference and create for the others
    - Proprietary format, not easy to parse
  - Define a very simple, neutral format (XML)
    - “Simple” to parse (tools existing)
    - Easy to extend if needed

SPI Software Distribution Service: Binaries

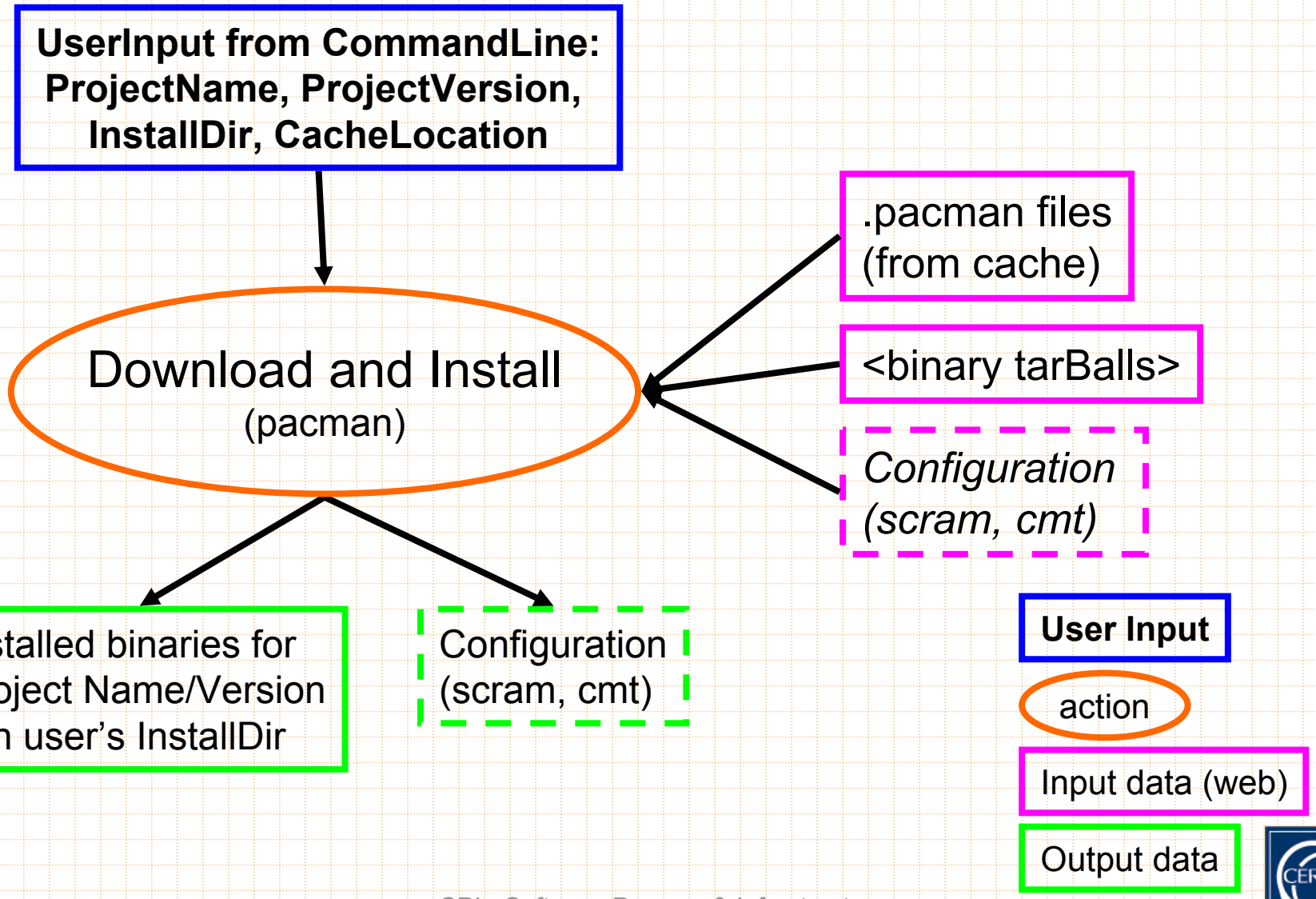
- **Simple solution to use**
  - Local installations (external sites, laptops,...)
  - Using simplest approach
  - Python downloader script + tar format
- **Distribution files are generated**
  - From general description of configuration (scram)
- **Simple tool to download and install**
  - successful for users installations
  - very easy to use and reliable
    - In use since 2003, few complaints

- **First version of Pacman binary cache available**
  - Feedback welcome

Binary installer (since 2003)



Request from expts: Pacman binary install



Software configuration and dependencies

- **XML description to have a simple to parse format**
  - ***Build information***
  - ***Dependency description***
- **Build and install for external packages and LCG software**
- **Aim: Generate the configurations for the build and release systems needed by the users**
  - SCRAM support
  - LCGCMT support
  - Config/make based solution
- **We don't want to make one more build and configuration system but generate for the existing ones**

Actions performed to fulfill the recommendation
Build System and Infrastructure (II)

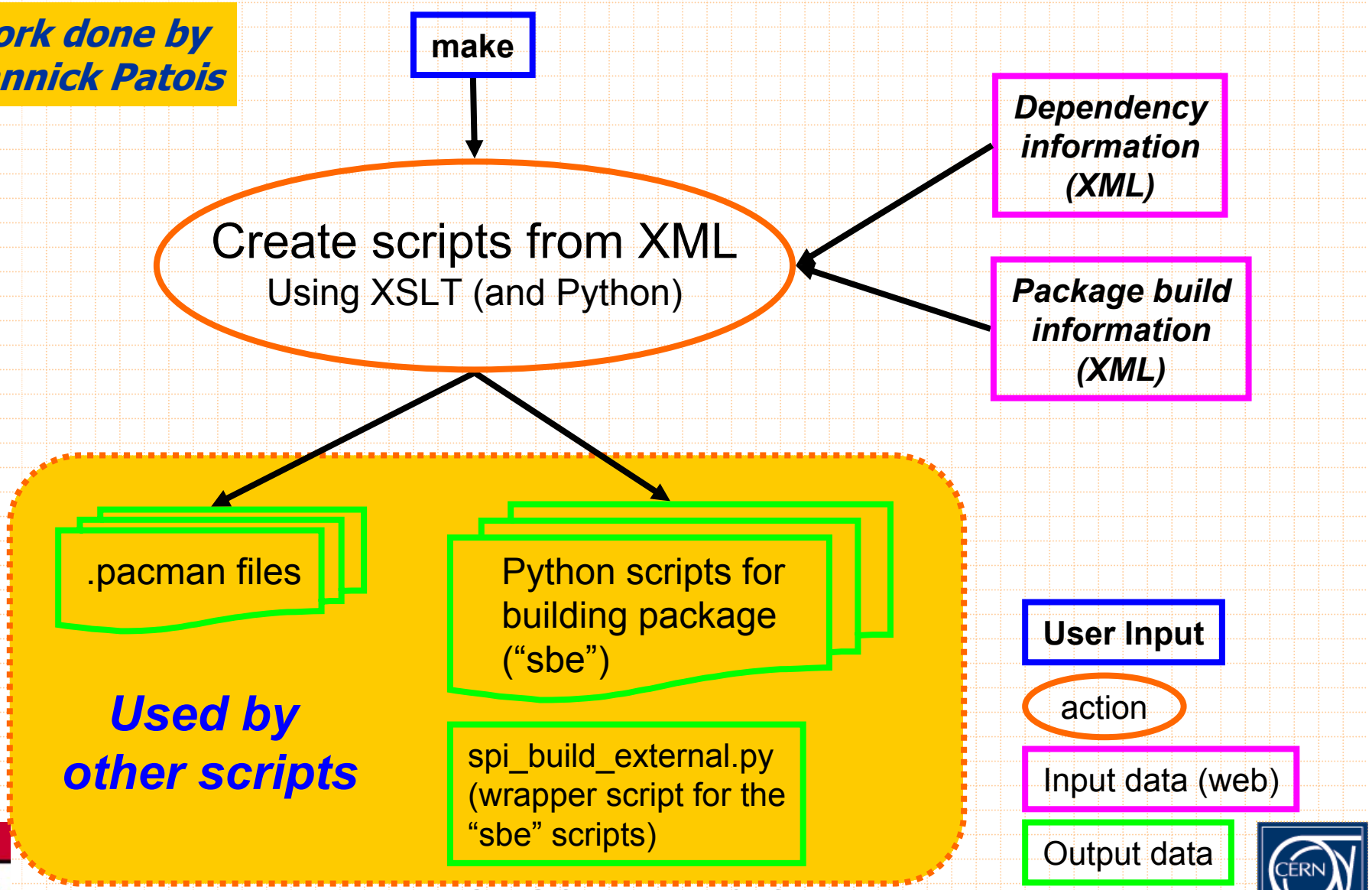
- XML description allows to support several build systems
- Generate and keep up to date configuration files for CMT
- Config/make solution is going to be achieved via the XML description files that he had to implement in order to support several tools for build and distribution

Software Distribution from Sources

- **Some packages have complex settings and/or dependencies**
  - Mysql, root, python
- **Need for consistent treatment of configure and compiler options across all external packages in use for project**
  - On project side this is done through scram
- **Need to accommodate several install/build systems**
  - Pacman, python scripts in use today
  - Rpm, apt, for the future ?
- **Decide to use independent XML format to describe the build process**
  - Use of XSLT (+ few python scripts) to transform this information into python scripts (“SPI build externals”, “sbe”) (Yannick Patois)
    - typically one per package/version

Core package: external package builder ("SPI build external" : "sbe")

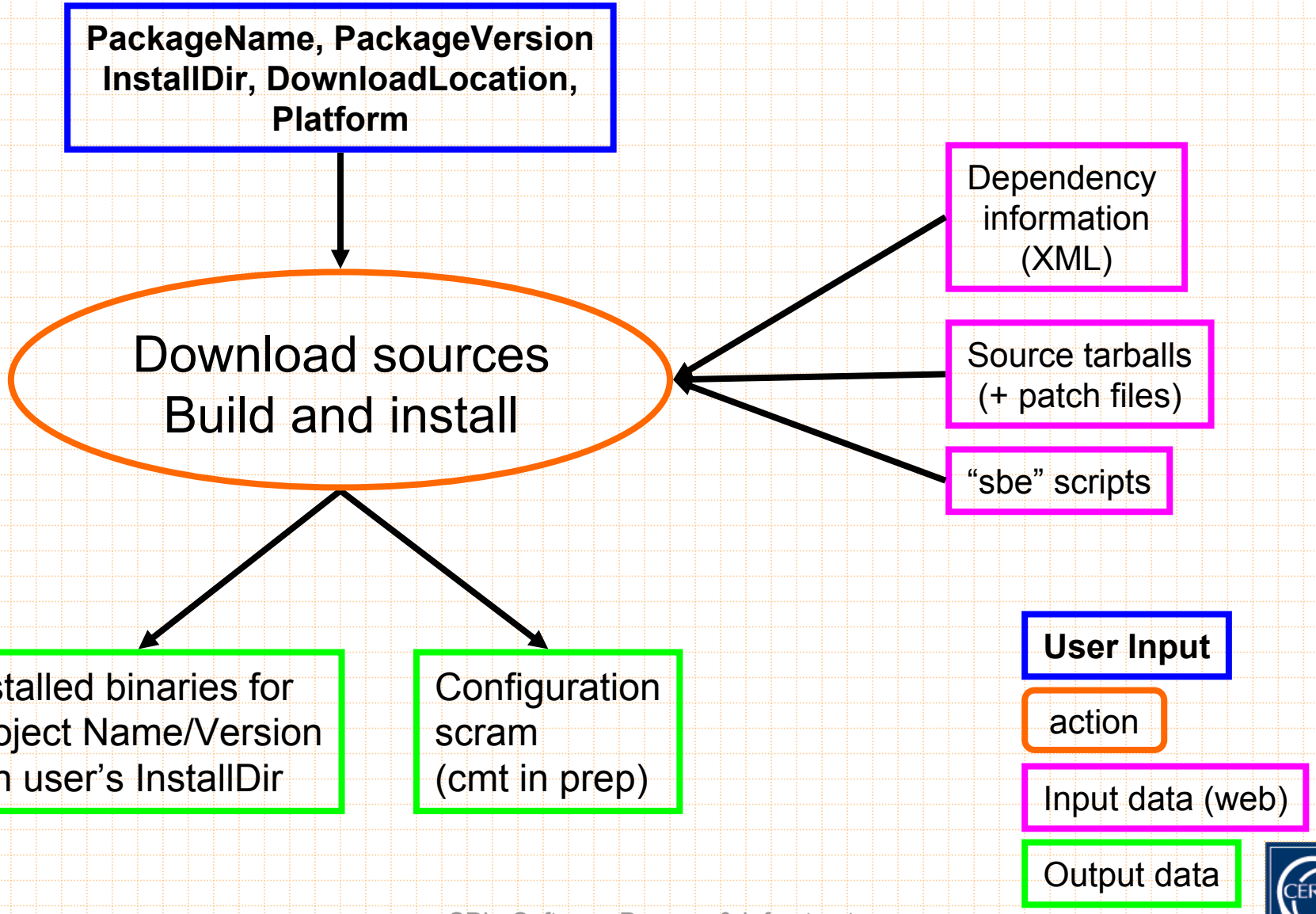
Work done by Yannick Patois



Software Distribution from Sources (II)

- **External treatment of dependency information (XML files)**
  - one per package/version, plus “meta packages”
    - e.g. boost-1.31.0.xml, external-lcg33.xml
  - Decouples build information from dependecy treatment
    - More flexible when dealing with installers
  - Python scripts resolves dependency hierarchy and provides build order
    - SPI configurator
- **Simple script to build all (“seti.py”)**
  - Resolve dependencies and get build order
    - SPI configurator
  - Call “sbe” for each external package neede
  - Build project(s) using scram “as usual”
  - First version ready
    - Documented in wiki: https://uimon.cern.ch/twiki/bin/view/SPI/SetiHowTo

Installation from sources



Actions Summary
Build System and Infrastructure (I)

- **Role of a central librarian**
  - Started in Summer 2004
- **Common build settings, release procedures**
  - Scripts for Linux, Mac
  - Different set of scripts for Windows (Pere)
- **Share of common build tools not developed within the projects**
  - Developed scripts to build and install project releases at CERN
    - Main difference from “outside”: several platforms
- **Build centrally the software and free developers in projects**
  - Done since late summer
- **Have prereleases available to the users**
  - Done. Users need to tell us when they don’t need the prerelease any more (to limit support requests to the projects)

Actions Summary
Build System and Infrastructure (II)

- **Investigate a config/make solution**
  - Postponed due to other (higher) priority issues
  - Do it inside the projects (with partitioning)?
- **Clarify long term strategy**
  - To be done (this review ?)
- **Support other build system used by other experiments (CMT)**
  - LCGCMT is distributed and maintained by SPI
- **Software Distribution**
  - More interaction with LCG deployment and check existing distribution tools (e.g. pacman)
    - Contacts with LCG deployment established but no common solutions adopted yet
    - Pacman has been investigated
      - Cache prepared for users
      - Problems using it internally
  - Customize distribution by platform and component
    - Ongoing work

Possible future improvements

- **Complete automation of procedures**
  - With documentation
  - Needed in “porting activities”
    - Gcc 3.4.3, 64 bit, Tiger, …
- **Provide working pacman caches**
  - Source and binaries
  - Feedback from expts/users needed
- ***Generate* configuration for scram and CMT**
  - from XML information
- **Provide other deployment formats used in expts ?**
  - Rpm, apt, … ?
  - Binary only or also from source ?

Collaboration with EGEE

- **Collaboration mostly with JRA1 (Middleware) and SA1 (Deployment) with considerable benefits. For all, I think.**
- **SPI provides tools for EGEE projects**
  - Presented in several occasions at all levels and activities
- **SPI services used**
  - Savannah portal used by several activities (Middleware, Deployment, etc). Requested several modifications to the software
  - Testing frameworks, all used (QMTest, CppUnit, PyUnit)
  - QA reports and metrics are being developed
- **SPI services not used**
  - No external software, no software distribution, no build system
- **Few (minor) specific services**
  - Verification of coding conventions via CodeWizard
  - dotProject service for basic planning
- **Contribution to SPI: 2 FTEs**
  Both very important for SPI
  - Yannick PATOIS (3/2004)
  - Johanne BENARD (8/2004)

→ Testing and Quality Assurance

Testing

- Current status
- Examples

Quality Assurance

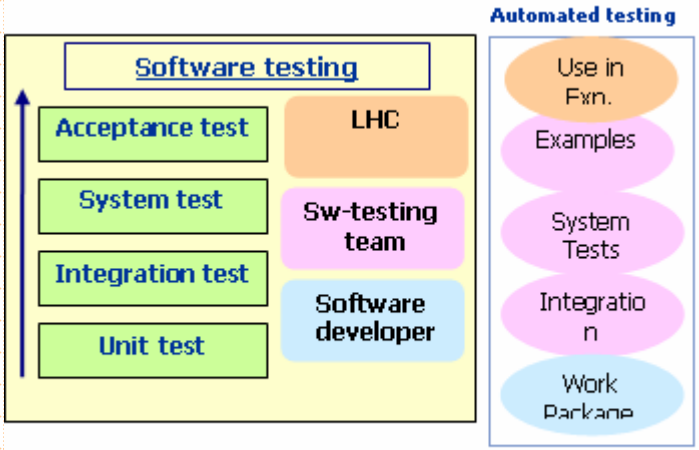
- Codewizard for EGEE
- QA reports
  - Savannah
  - Test coverage
- Future actions

SPI Testing services

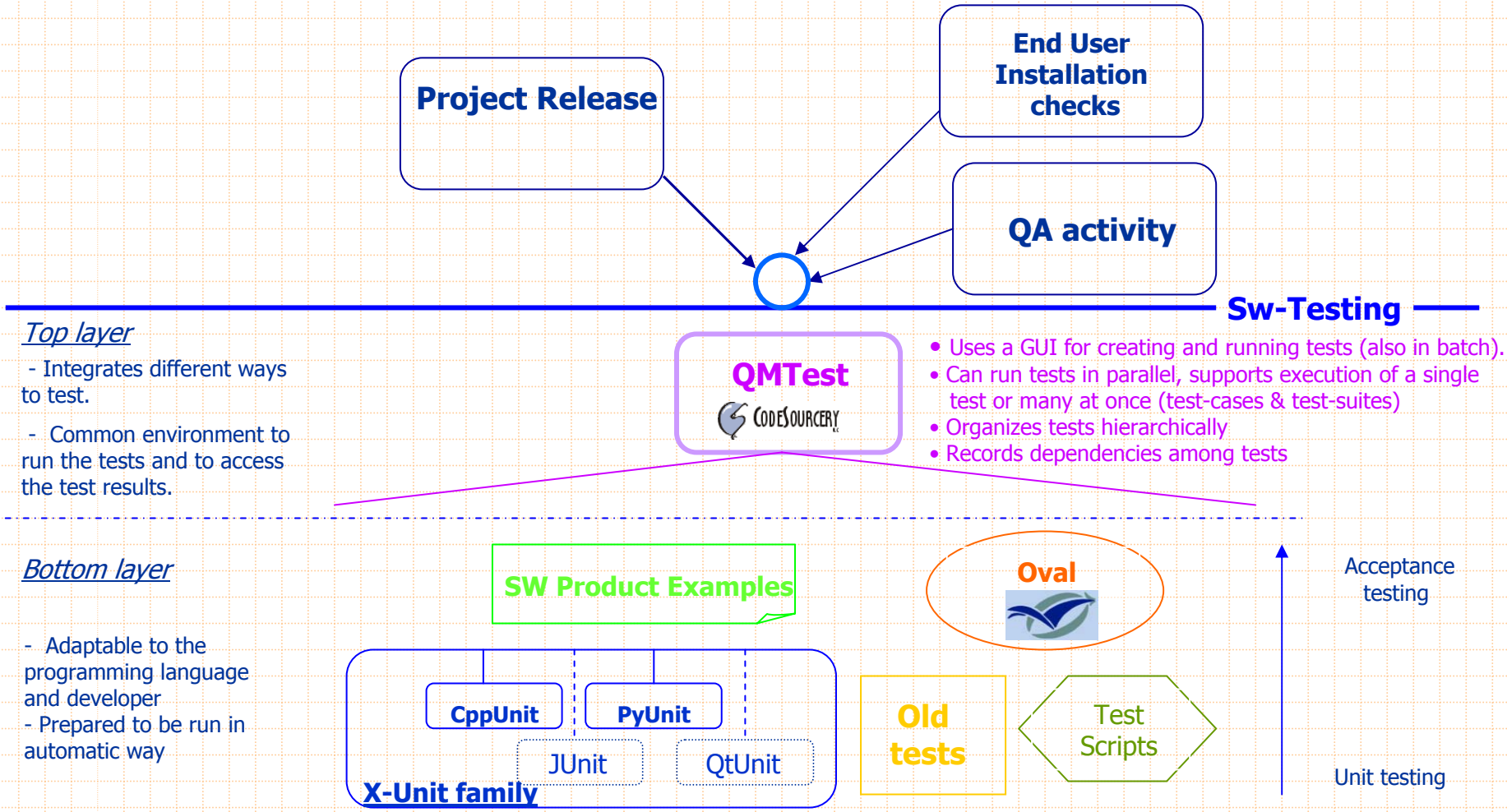
- **Testing should be an integral part of the software development in the LCG App Area**
- **Goal was to provide something that can be run automatically as often as needed (development, release, etc)**

SPI provides

- **Test frameworks**
  - CppUnit, Oval
  - Qmtest
- **Test Support**
- **Test policies**
- **Test document**



Testing framework – global picture



Testing other services

- **Support**
  - Web information

LHC Computing Grid > LCG App Area > SPI Home

SPI Quick Links

SPI Home
SPI Index Page

SPI Workbook

SPI Services Links

LCG Workbook

Savannah Portal
External Software
Software Testing

Software Download
Quality Assurance

LCG App. Area

Home Page
LCG Agenda

PI Project
POOL Project
SEAL Project
Simulation Project
SPI Project

External Links

CERN
EP Division
IT Division
LCG
EGEE

LHC experiments

ALICE
ATLAS
CMS
LHCb

LCG Application Area - LCG Infrastructure: SW - Testing

HowTo for CppUnit TestFramework

What is CppUnit?
How to start testing with CppUnit
How to use CppUnit in LCG AppArea
LCG CppUnit FAQ
Related information

What is Cppunit?

CppUnit is a framework for writing unit tests in C++ and running them automatically, giving a report about success or failing tests. It is the C++ port of the famous JUnit framework for unit testing. CppUnit is one of the members of the "XUnit family" (JUnit, PerlUnit, PyUnit, QtUnit, ...) test frameworks free available from XP software. The test output is in XML or in text format for automatic testing and GUI based for supervised tests.
The first port of JUnit to C++ was done by Michael Feathers. His versions can be found on the XProgramming software page. They are os-specific, so Jerome Lacoste provided a port to Unix/Solaris. His version can be found on the same page. The CppUnit project has combined and built both on this work. The main purpose of CppUnit is to support developers in doing their unit testing of C++ programs.

CppUnit works in Linux, Solaris and Windows platforms. See the SPI supported platforms+compilers at SPI external software service.

return to top of page

How to start testing with Cppunit

Installation check

To use CppUnit the CppUnit libraries and include files should be installed or accessible to your machine. To check this point in a linux/unix machine you can type at the prompt

```
$ cppunit-config [--version] [-cflags] [--libs] [--prefix] [--help]
```

Using the different optional flags you can inquire the system about the CppUnit installed version, the cflags and library path you have to use, where CppUnit was installed and this help.
If you are using one of the initial shell scripts provided by SPI in the general HowTo for Test Execution Frameworks all the needed information is loaded in the environment variables ($PATH, $LD_LIBRARY_PATH, $CPPUNIT_LDFLAGS, $CPPUNIT_INCLUDES, $ACLOCAL) that you can check at the prompt with the echo

data along with the expected results for a particular test objective

Testing other services (2)

- **Support**
  - Web information
  - How-To
- **Policies**
- **Documents**

LHC Computing Grid > LCG App Area > SPI Home

SPI - Software Process & Infrastructure

Software Testing Policies

- ***Design phase*** *(for Sw-testing team within a project):*
  - Perform a Testing Planning
- ***Code phase*** *(for developers within a project):*
  - Populate, document and execute Unit Test cases
  - Populate and document Integration-Tests and System-Tests
  - Perform the bug tracking and convert into tests all those fixed bugs susceptible to be use in the regression testing scheme
- ***Test phase*** *(for SW-testing team within a project)*
  - Conduct unit, integration and systems tests
  - Perform regression testing as needed
  - Conduct acceptance tests (alfa tests)
- ***Test environment and tools***:
  - Tests must be run in automatic mode for each release, pre-release and nightly building using the test frameworks and tools listed below:

    QMTest
    Oval
    CppUnit, PyUnit... X-Unit

  - The complete set of test must be run through QMTest already integrated with NICOS nightly building system.
  - Test-plan and Test-case documentation must be done using the proposed templates.
  - The use of any other test framework, scripts, tools and improved documentation must be discussed with SPI in benefit of all LCG AppArea projects.
- ***Sw-testing QA check list:*** the QA activity will verify if the projects are compliant with LCG Application Area Policies and the check list of assessed items is available.

Testing usage in LCG

- All projects are using one of the X-Unit family: CppUnit PyUnit, …
- POOL is using OVAL
- All projects are using the Qmtest test framework

- A few figures:

	PI_1_3_0	POOL_2_0_3	SEAL_1_6_1
Unit tests	93	101	308
Integration tests	24	64	13
Total number of tests	117	165	321

Testing usage in EGEE

- **One of the groups to install the SPI testing frameworks is the EGEE integration team: CppUnit, PyUnit, Qmtest as web server.**

EGEE Home | Intranet Home | Search | EDMS Documents | People | Calendar | Agenda maker | Glossary

egee
Enabling Grids for E-science in Europe

JRA1: Software Testing

Tools | Testing | Integration | Information Services | Workload Management | Data Management | Security | Management

JRA1 Home | Mandate | People | Meetings | Presentations | Mailing Lists | Useful links | Test Plans | Action lists | EDMS | Agenda maker | Savannah portal

TESTING ACTIVITIES

MJRA1.3 JRA1 Test Plan

Testing and validation testbed: schematic

Functional Test reports:

Installation testing: Test results

WMS testing knowledge database: WMS knowDB

Test Templates: Test Case Template

REQUIREMENTS AND DESIGN DOCUMENTS

HEP: HEPCAL; HEPCAL II; AWG Use cases; AWG Recommendations;

ARDA: ARDA Final Report,

Biomedical and Generic: NA4 requirements database; Medical Data Storage and Access requirements; Bio-informatics Requirements (D10.1)

PTF: Central database prioritized by the PTF

EGEE: Architecture document; Design document

News

- Testing coordination meeting scheduled for 7 September 2004 at CERN
- 01/07/2004: All testbed machines replaced and being reinstalled
- Testing Web Site Launched
- RAL and NIKHEF official external testing sites

.... Previous news

TESTING TOOLS

Testing tools survey

LCG Twiki database for debugging problems

Requirements for testing tools for EGEE

Junit - CPPUnit - PyUnit - QMTest - NMI Condor

Related Documents

JRA1 Testing Workload Breakdown plan

dotProject for the testing activity

Glossary of testing terminology

Metrics for the testing activity

Access to restricted area:
Members Area

Codewizard

- **Integration of Codewizard to EGEE middleware building process**
- **Codewizard is a C and C++ rule checker highly customizable:**
  - Effective C++
  - More effective C++
  - Meyers-Klaus
  - Universal Coding Standard
  - Other
- **Output example**

```
/scratch1/testcwprod/egeewkspc/org.glite.wms.common/src/configuration/ModuleType.cpp
 In ModuleType.cpp, at line 20
 Error of category:    Prefer C++-style casts

/scratch1/testcwprod/egeewkspc/org.glite.wms.common/src/configuration/ModuleType.cpp
 In ModuleType.cpp, at line 22
 Error of category:    Prefer C++-style casts
```

Codewizard (2)

- **More information on SDT:**

PARASOFT

Codewizard 4.3

Version 4.3 is now available

The key to simultaneously reducing development time while increasing software quality is to prevent errors at the beginning stages of development. One of the most effective ways to prevent such errors is to follow coding guidelines. CodeWizard, an advanced C/C++ source code analysis tool, uses over 300 industry-respected coding guidelines to automatically identify dangerous coding constructs that compilers do not detect. CodeWizard makes it easy to create new, customized rules through the RuleWizard feature, or to suppress rules for customized analysis. When used on a daily basis, CodeWizard simplifies code reviews and make code more readable and maintainable.

Use
Linux
Solaris
Windows
Information
Licencing
Documentation
Examples
Known Issues

Benefits

CodeWizard is the ideal foundation for implementing team and organization coding guidelines. A wide variety of platforms and command line functionality enable your development team to start using coding standard guidelines immediately. Additionally, CodeWizard:

- Results in faster time-to-market
- Improves application reliability
- Reduces overall cost of development, support, and maintenance
- Speeds 32-bit to 64-bit porting, improving reliability of the new 64-bit code
- Accelerates C and C++ learning curve
- Speeds up code reviews
- Improves code readability

SPI QA services

- **QA activity main goal: help LCG projects to assess and to improve their software and procedures quality**

QA review

- **Fully automated QA**
- **Resource centrally responsible for QA**
- **Policies VS Tools to help facilitate project compliance**

QA procedures provided:

- **QA checklist on each release**
  - Build the release
  - Run automatic tests
  - Statistics
- **LCG Policies**
- **QA automated report**
  - QA reporting on released project
  - Savannah QA reporting
  - Test Coverage

Quality Assurance Web (http://spi.cern.ch/qa)

LHC Computing Grid > LCG App Area > SPI Home

SPI - Software Process & Infrastructure

Updated
01-Nov-2004 15:51

SPI Quick Links

SPI Home
SPI Index Page

SPI Workbook

SPI Services Links

LCG Workbook

Savannah Portal
External Software
Software Testing

Software Download
Quality Assurance

LCG App. Area

Home Page
LCG Agenda

PI Project
POOL Project
SEAL Project
Simulation Project
SPI Project

External Links

CERN
EP Division
I T Division
LCG
EGEE

LHC experiments

Quality Assurance

The main goal of QA activity is to **help projects assess and improve the quality of their software and procedures**.
This means among others:

- verify if project setup is correct and compliant with LCG Application Area Policies;
- provide tools to collect useful software metrics which help to asses software quality;
- provide monitoring tools to see the evolution of the projects.

By applying:

- Clear rules and the checklist of assessed items is available here. This list is known in advance to projects.
- QA Reports are generated automatically by the tools and contain data and statistics about the project, so the reader may easily track the project evolution.
- Everybody who is interested may see (and generate) the report at any time for any release of any project.

Related Documents:

- LCG QA Checklist
- LCG Application Area Policies

Interactive QA report generation

Find the Web form here.

QA Savannah reports	1. EGEE groups: o JRA1 Middleware: bugs by "Category" o JRA1 Coordination: bugs by "Status" 2. LCG groups: o LCG2 sites: tasks by "Assigned To" o LCG2 sites: tasks by "Submitted By" o SEAL: bugs by "Category" o POOL: bugs by "Category" o PI: bugs by "Status" o SPI: bugs by "Category" 3. CMS groups:

QA checklist

LHC Computing Grid > LCG App Area > SPI Home

SPI - Software Process & Infrastructure

Updated 12-Nov-2004 16:26

SPI Quick Links

SPI Home
SPI Index Page

SPI Workbook

SPI Services Links

LCG Workbook

Savannah Portal
External Software
Software Testing

Software Configuration

Software Download
Quality Assurance

LCG App. Area

Home Page
LCG Agenda

PI Project
POOL Project
SEAL Project
Simulation Project
SPI Project

External Links

QA Checklist

italics mark the point with less priority

1. **Building the release**
   1. build a complete release following the standard sequence of commands
      ```
      scram project PROJECT PROJECT_x_y_z
      cd PROJECT_x_y_z
      cvs -d :pserver:anoncvs@lcgapp.cern.ch:/cvs/PROJECT co -r
      PROJECT_x_y_z -d src project
      eval `scram runtime -sh`
      cd src scram build
      ```
   2. run automatic tests with qmtest and get the output, standard sequence of commands: [DONE]
      ```
      scram project PROJECT PROJECT_x_y_z
      cd PROJECT_x_y_z
      cvs -d :pserver:anoncvs@lcgapp.cern.ch:/cvs/PROJECT co -r
      PROJECT_x_y_z -d src project
      eval `scram runtime -sh`
      qmtest -D ./src/config/qmtest run -o test_results.qmr
      ```
2. **Statistics**
   1. Test Inventory
      1. number of tests: per package, other tests (integration)[DONE]
      2. number of tests in automatic test system [DONE]
      3. number (and names) of tests which failed (for automatic test system) [DONE]
      4. *memory checker report for automatic tests in case of memory leaks*
   2. Documentation Inventory
      1. number of (meaningful) warnings from automatic refdoc tool per package
      2. number of packages without user documentation
   3. Examples Inventory
      1. number of examples: per package, per project
   4. Savannah Statistics [DONE]
      1. number of users (who submitted at least one bug) [DONE]
      2. number of bugs posted [DONE]
      3. number of opened bugs [DONE]
   5. Code Inventory

QA - LCG policies

LHC Computing Grid > LCG App Area > SPI Home

SPI - Software Process & Infrastructure

Updated
12-Nov-2004 16:26

SPI Quick Links

SPI Home
SPI Index Page

SPI Workbook

SPI Services Links

LCG Workbook

Savannah Portal
External Software
Software Testing

Build servers
Software Configuration

Software Download
Quality Assurance

Software Development Tools

Setting-up the environment.

LCG software is compiled using SCRAM. See SCRAM Manual.

Tools for automatic generation of source files and directories.

Software Development Frequently Asked Questions

Click here to access the FAQ list.

LCG Software Development Policies

Click here to access the LCG policies page

QA reporting

- **QA reporting on project released**
  - In place, is in standard LCG environment under /afs/cern.ch/sw/lcg/app/spi/tools/latest
  - Analyze project tree in AFS release area
  - Analyze code
  - Run tests

QA – code statistics

QA Report for GENSER_0_3_0

```
Created: Wed Mar 23 18:24:53 2005 by jbenard
Command: - --no-doc GENSER_0_3_0
Option values used to create the report:
Title=QA Report for GENSER_0_3_0
Project name=GENSER_0_3_0
Documentation not required
```

Quality Assurance Home Page

Table of Contents

1. Source Code Statistics

Back to Table of Contents

GENSER LXR page

Package	nFiles	TotalNL	SLOC	meanSLOC	sigSLOC	maxSLOC	comment
1_0	199	27979	43620	219	313	3132	
1_1	199	27487	42979	216	314	3128	
Total	398	55466	86599	217.5854271356784			

2. CVS Structure

QA – test statistics

QA Report for SEAL_1_6_1

```
Created: Wed Mar 23 18:44:09 2005 by jbenard
Command: - --no-doc --run-tests SEAL_1_6_1
Option values used to create the report:
Title=QA Report for SEAL_1_6_1
Project name=SEAL_1_6_1
Documentation not required
Tests run
```

Quality Assurance Home Page

Table of Contents

Number of tests performed:

	Total	Failed
UNIT_TESTS	287	56
INTEGRATION_TESTS	0	0
TOTAL_TESTS	287	56

--- STATISTICS ----------

287 tests total
34 (12%) tests FAIL
22 (8%) tests UNTESTED
231 (80%) tests PASS

QA - Documentation

Software Process Infrastructure

Jump:

SPI

SPI Wiki
SPI Wiki Home
Changes
Index
Search

SPI Web Home

Other Wikis

ADCgroup
Atlas
CMS
CS
Controls
DbaServices
Know
Main
PSgroup
SDT
SPI
Sandbox
TWiki

Create personal sidebar

Edit Attach Printable SPI.HowToQAReports r1.3 - 23 Mar 2005 - 17:48 - Main.jbenard topic end

Back to SPI Workbook

How to produce QA report

QA reports may be produced either by command line.

By command line:

NB: an access to AFS is required

- Go to

  `/afs/cern.ch/sw/lcg/app/spi/tools/latest/setup`

- Set the environment by sourcing lcgspi.csh or lcgspi.sh script: C Shell "source setup/lcgspi.csh", Bash Shell "source setup/lcgspi.sh"
- Go to /afs/cern.ch/sw/lcg/app/spi/tools/latest/scripts
- Launch the program by command line, see the Option List section in this page.

  To see help : lcg-qa-project-report --help

Options list

Several options are available to the user:

1. **Project:**
   States the project name (any of the project release on AFS see /afs/cern.ch/sw/lcg/app/release).
   The project name could be provided with or without a version number E.G. SEAL_1_1_0 or SEAL
   If project=SEAL_1_1_0, a QA report for SEAL release=1_1_0 will be produced.
   If project=SEAL, a QA report for the last released version on AFS will be produced.

QA Savannah

- **From a web form**
- **Available for anybody from SPI QA web**
- **New features:**
  - Begin &end date,
  - Tracker (Bug, Task, Support, Patch)
  - Savannah field,
  - Style sheets (header, title)
  - Configuration file
- **Examples**

QA Web Form

LHC Computing Grid > LCG App Area > SPI Home >

SPI - Software Process & Infrastructure

Updated
07-May-2004 16:27

SPI Workbook

LCG Workbook

SPI Quick Links

SPI Home
SPI Index Page

SPI Workbook

SPI Services Links

LCG Workbook

Savannah Portal
External Software
Software Testing

Software Download
Quality Assurance

LCG App. Area

Home Page
LCG Agenda

PI Project
POOL Project
SEAL Project
Simulation Project
SPI Project

Savannah QA tracker analyser

Please, read the How To page before using this form.

Tracker:

Bugs

Main options:

Project Name

Example of project name: POOL, JRA1 Middleware
Find the project name list..

Start date (DD/MM/YYYY) **End date** (DD/MM/YYYY)

Configuration file:

1. Please either use the project configuration file, stored on AFS (/afs/cern.ch/sw/lcg/app/spi/qa/config/savannah)
   Use **project configuration file** ☐
2. **Or**, fill in the following fields,
   Savannah Field
   (example: Category, Severity, Assigned To ...)

Submit Query | Reset

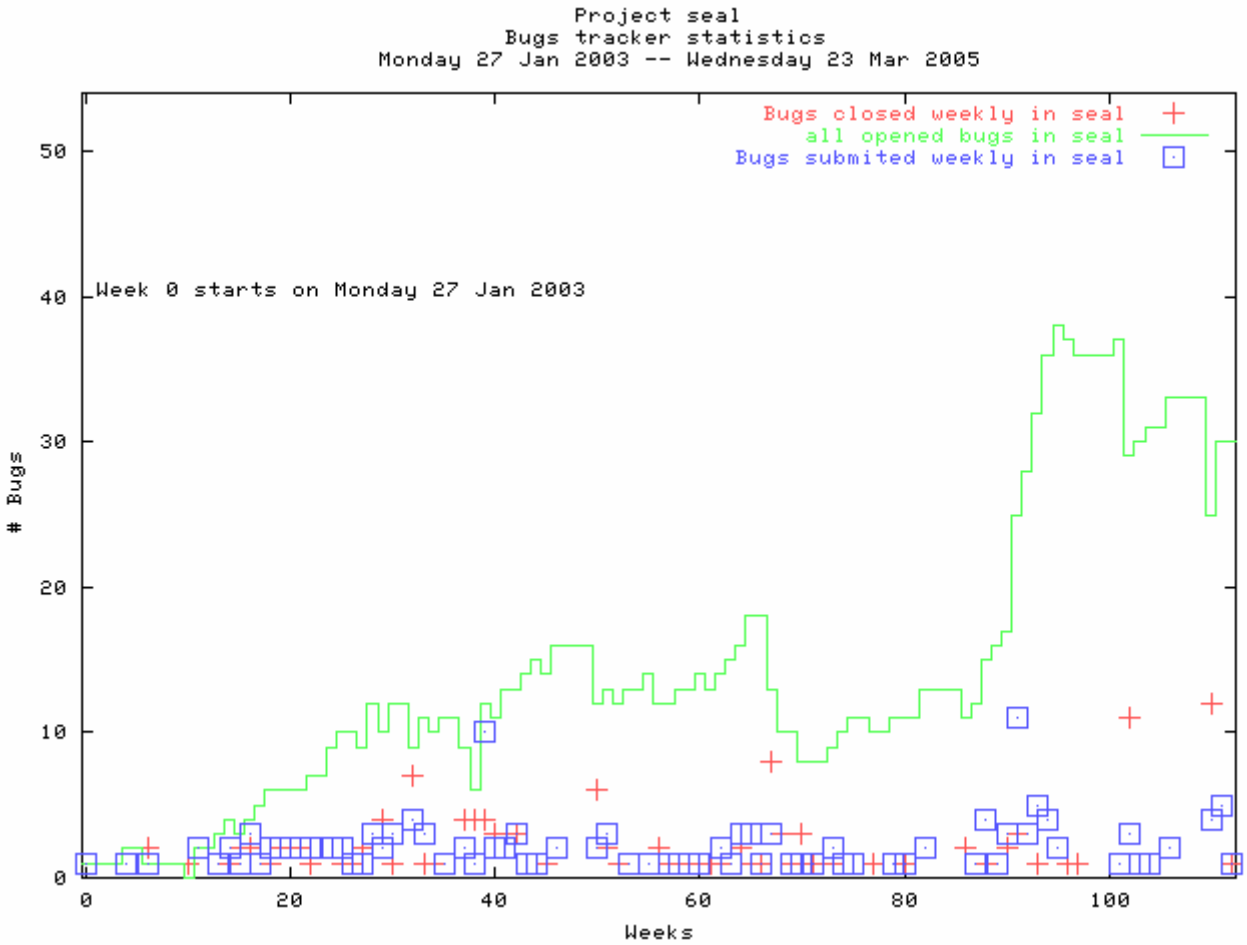
Example

LCG QA Report for SEAL (seal)

```
Created: Tue Mar 29 14:18:21 2005 by apache
Command: - --config-file=/afs/cern.ch/sw/lcg/app/spi/qa/config/savannah/seal/QA_report_config_file.ini
--tracker=bugs --no-doc --only-savannah seal
Option values used to create the rep
Title=LCG QA Report for SEAL (seal)
Header file=/afs/cern.ch/sw/lcg/app/
Label=category_id
Project name=SEAL (seal)
Only Savannah section is performed
Documentation not required
```

Quality Assurance Home Page

Table of Contents



Example (2)

Software Process Infrastructure

EGEE Home | Intranet Home | Search | EDMS Documents | People | Calendar | Agenda maker | Glossary

egee Enabling Grids for E-science in Europe

Test for LCG REVIEW: JRA1 Middleware (jra1mdw)

Tools	Testing	Integration	Information Services	Workload Management	Data Management	Security	Management

JRA1 Home | Mandate | Meetings | Presentations | Savannah | EDMS

```
Created: Tue Mar 29 14:38:23 2005 by apache
Command: - --config-file=/afs/cern.ch/sw/lcg/app/spi/qa/config/savannah/jra1mdw/QA_report_config_file.ini
--tracker=bugs --no-doc --only-savannah jra1mdw
Option values used to create the report:
Title=Test for LCG REVIEW: JRA1 Middleware (jra1mdw)
Header file=/afs/cern.ch/sw/lcg/app/spi/qa/config/savannah/QA_EGEE_header.html
Label=category_id
Project name=JRA1 Middleware (jra1mdw)
Only Savannah section is performed
Documentation not required
```

Quality Assurance Home Page

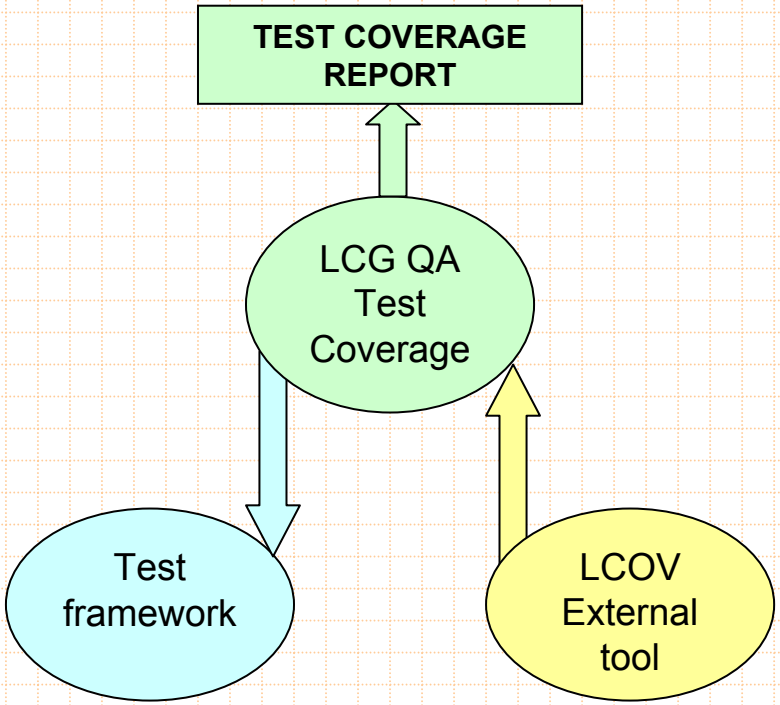
Table of Contents

LCG

CERN

Test coverage

- Under development (not public yet)
- Should be available for any project member (developers, release manager, etc)
- Assess the percentage of executable lines reached by the tests



Test coverage example

LTP GCOV extension - code coverage report

LTP GCOV extension - code coverage report

Current view: directory - src/MultiCollection/src
Test: lite_test_coverage_POOL_2_0_0.info
Date: 2005-03-23
Code covered: 77.9 %

Instrumented lines: 235
Executed lines: 183

Filename	Coverage		
IMultiCollection.cpp		54.4 %	37 / 68 lines
MC_DeepObjCollIterator.cpp		85.7 %	48 / 56 lines
MC_DeepObjCollIterator.h		80.0 %	4 / 5 lines
MC_DeepObjIterator.cpp		94.7 %	18 / 19 lines
MC_DeepObjIterator.h		0.0 %	0 / 1 lines
MC_FlatCollIterator.cpp		84.5 %	49 / 58 lines
MC_FlatCollIterator.h		0.0 %	0 / 1 lines
MySQLMultiCollection.cpp		100.0 %	12 / 12 lines
RootMultiCollection.cpp		100.0 %	12 / 12 lines
modules.cpp		100.0 %	3 / 3 lines

Generated by: LTP GCOV extension version 1.1

Future improvements

- QA savannah reporting from Savannah export
- Test coverage available also for internal releases
- Integration with post release actions
- Documentation and work with the projects
- Automation of tool is going on

Milestones (I)

2004/1/31	Done	IT CVS service verified and validated by SPI
2004/1/31	Done	More code standards checks added to the QA reports (via doxygen)
2004/2/20	Done	Upgrade of the Savannah service and installation of the Gnu open source version
2004/2/28	Done	QA reporting tools available publicly
2004/2/28	Done	Delivery of configuration files also for the CMT build system
2004/3/15	Done	LCG software librarian in place
2004/4/1	Done	Migration of all projects to IT CVS service
2004/4/15	Done	Certification of external software for the new Linux platform

Milestone (II)

2004/5/1	Done	Definition of the EGEE collaboration
2004/5/15	Done	RH 7.3 gcc 3.2.3 supported
2004/7/1	Done	Common build and release solution in LCG App Area
2004/10/15	Done	Naming and coding conventions for EGEE Middleware in the build system
2005/10/31	Done	Adapt the QA reports to EGEE QA standards
2004/11/15	Done	Pacman cache for binaries of external and LCG packages
2004/11/20	Done	Definition of software configuration information in XML
2004/11/30	Done	New release of Savannah

Milestones (III)

2004/12/1	Done	QA reports in production as deliverable of the LCG projects
2004/12/15	Done	Sources distribution and build of external and LCG software
2004/12/15	Done	SPI Librarian performs the releases and pre-releases of the LCG software
2005/1/15	Done	Definition of build information in XML
2005/1/15	Done	QA Tests Coverage reports in production
2004/12/15	OTW	Definition of the products and partitions of the LCG software, for individual partial releases and bug releases
2005/2/28	OTW	Independent build and release of partitions of the LCG software

Unplanned Milestones and Activities in 2004

- **We have many services running and using all resources assigned, we follow the AF's decisions for our priorities and activities**

- **The certification of the new CERN platform and the adoption first of CEL3 and later of SLC3 required more work than expected**
  - SPI had to compile and provide several versions of the external packages in order to accommodate the needs of different experiments and to suit their software configurations.
- **Support for Mac OSX, SLC3 and RH73_gcc323 were decided as priority**
  - Delayed the port to other platforms (such as icc 8);
- **We had to start defining a simple XML description of build/config info**
  - This will simplify generation of specific information needed by several external tools (Scram, CMT, pacman, etc)
  - Required investing more time that is saved when a new installation is needed
- **EGEE collaboration started successfully**
  - 2 FTE contributed to the SPI project
  - SPI was involved in presenting/adapting its existing services. In particular the collaboration is the Middleware (JRA1) and Quality Assurance (JRA2) activities.
  - The services involved were Savannah, QA and Testing in particular.
- **Savannah service required the implementation of new features to suit LCG and EGEE needs.**

Summary

- **SPI provides a set of stable services that fully use the current resources**
  - Savannah, External Software, Testing Frameworks, Software Distribution, Build and Release, QA Activities
- **Resources will be reduced at the end of 2005 but we will not reduce the current services**
  - We will maintain and automate the existing ones
  - No requests for major new services from LCG, need to adapt or interface to external tools
- **We refer to and follow the guidelines of the Users**
  - Represented at the Architects Forum and steering our priorities
- **Whenever possible we also do help/work for experiments specific needs**
  - Installations of additional software, interface to their build/distribution systems, add features

Proposed focus for 2005

- Continue to automate and centralize the cycle for software externals/build/test/release/distribution
- Provide software that can be installed or downloaded on several platforms using established tools
- Increase work on the QA and testing activities in the projects
- Focus more on encouraging other areas and projects to use uniform solutions provided
- **Further priorities and milestones will be defined according to the recommendations of the review**

SPI would like to thank…

- **Release managers in the projects**
  - L.Moneta (Seal, Pi), I.Papadopulos (Pool)
- **Librarians and Integrators in the LHC Experiments**
  - F. Ranjard, S.Muzaffar, E.Obreshkov, D.Quarrie
- **For their important tools and friendly help**
  - M.Roy (Savane), S.Youssef (Pacman)
- **QMtest setup in the projects**
  - G.Govi (Pool), L.Moneta (Seal)
- **Doxygen scripts and testing**
  - A.Zsenei(Seal)
- **Former SPI people that helped when there was nobody in QA and testing**
  - M.Gallas, J.Moscicki
- **All the developers in the experiments and projects for their feedback on each service**

BACKUP MATERIAL

- **BACKUP MATERIAL**

Resources

Services	Responsible	%
External software, installation and distrib.	**E.Poinsignon**	**75**
Savannah service, bug and devel. portal	**Y.Perrin**	**100**
LCG Librarian, Build and Configuration	**A.Pfeiffer**(5/04)	**70**
Documentation, Workbook, Policies, Web	**A.Aimar**	**70**
Build tools, Software Distribution	**Y.Patois** (EGEE 4/04)	**100**
QA reports and Testing Frameworks	**J.Benard** (EGEE 8/04)	**100**