Computing in High Energy and Nuclear Physics
Prague | Czech Republic | 21 - 27 March 2009

# Organization, Management, and Documentation of ATLAS Offline Software Releases

Presented by Frederick Luehring
Indiana University
On Behalf of the ATLAS Software Infrastructure Team (SIT)

Computing in High Energy and Nuclear Physics
Prague | Czech Republic | 21 – 27 March 2009

## Authors

ALBRAND, Solveig (Laboratoire de Physique Subatomique et de Cosmologie (LPSC)), AMRAM, Nir (Tel Aviv University), BLACK, Kevin (Harvard University), CIBA, Krzysztof (AGH-UST, Kraków), DE SALVO, Alessandro (Universita & INFN, Roma I), FULACHIER, Jerome (Laboratoire de Physique Subatomique et de Cosmologie (LPSC)), GALLAS TORREIRA, Manuel (CERN), HAYWOOD, Stephen (Rutherford Appleton Laboratory), JAIN, Vivek (Indiana University), KACHAEV, Igor (Institute for High Energy Physics (IHEP)), LAMBERT, Fabian (Laboratoire de Physique Subatomique et de Cosmologie (LPSC)), LLOYD, Steve (Queen Mary University of London), LUEHRING, Frederick (Indiana University), MOYSE, Edward (University of Massachusetts), OBRESHKOV, Emil (Deutsches Elektronen-Synchrotron (DESY)), PACHECO, Andreu (IFAE Barcelona and CERN), QUARRIE, David (Lawrence Berkeley National Lab. (LBNL)), RYBKINE, Grigori (Lab. de l'Accelerateur Lineaire (IN2P3) (LAL)), SHERWOOD, Peter (University College London), SIMMONS, Brinick (University College London), THOMPSON, A. Stanley (University of Glasgow), UNDRUS, Alexander (Brookhaven National Laboratory (BNL)), VON DER SCHMITT, Hans (Max-Planck-Institut fuer Physik), YOUSSEF, Saul (Boston University), ZENIN, Oleg (Institute for High Energy Physics (IHEP))

Disclaimer: The work presented was done with the contributions of many people!

# Collaborative SW Building

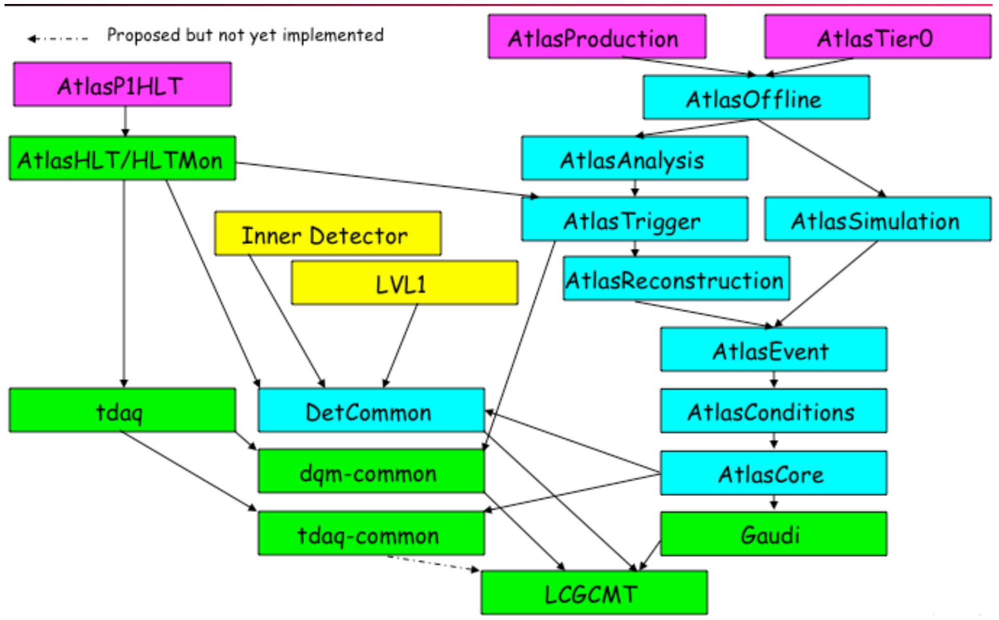
- There are currently just over 800 people registered as ATLAS offline software developers and about 400 active developers.
  - These developers are located in ~40 countries world-wide and using their code, ATLAS must build timely, working software releases.
  - To this date, ~7 Million Lines of Code in ~3000 packages have been written mainly in C++ and Python (also Fortran, Java, PERL, SQL).
    - The main platform is Linux/gcc (SLC4 soon to be SLC5)
- The ATLAS Software Infrastructure Team (SIT) provides the infrastructure used by these developers to collaboratively produce many versions of the ATLAS software.
  - There have been 15(!) major releases of the ATLAS SW since May of 2000 and hundreds of smaller releases.
- Note: ATLAS shares code between the online (Trigger/DAQ) software and the offline software.
  - This talk is mostly about the offline portion of the software.

# Membership of the SIT

- The ATLAS Software Infrastructure Team is made up of ~30 people.
  - There are a few people working full time on the SIT activities but most people are volunteers working a small percentage of their time.
  - Many of the SIT members are computing professionals with a few physicists & technical physicists included.
    - Those ~30 people amount to ~12 Full Time Equivalents (FTEs)
  - Just like the developers, the SIT members are spread out worldwide with a core of about 4-5 people at CERN.
  - Most SIT members receive only service work credit (and the satisfaction of doing a necessary job).
  - Quarterly meetings, bi-weekly phone calls and huge numbers of emails are used to coordinate the work.
- Organizing this work to be effective is crucial given its importance and the diverse nature of the SIT. The SIT coordinator bears responsibility for keeping all of the people and all of the jobs organized.

## Software Versions/Diversity

- In addition to the large number of developers and the large amount of code, the SIT is supporting several versions of the software because of requests from different communities within ATLAS.
  - E. g. Simulation, Reconstruction, Cosmic Running, MC production, etc.
  - This puts a large stress on the SIT because often (usually!) priorities conflict between the various groups.
  - Various meetings are held to define the priorities.
- To aid in managing this diversity, the software is partitioned into ~10 offline SW projects (cyan on diagram).
  - This allows the SIT to separate work flows and to control dependencies.



## Release Strategy

- We have had to significantly modify our release building strategy.
  - Previously we had nightly builds every night, development builds every month, production releases every six months, and bug fix production releases as needed (as reported at CHEP06 in Mumbai).
    - Typically there were at most two releases operating at one time: one for fixing bugs in the current production release and one that was developing new functionality for the next production release.
  - Now it is much more complex: ~35 releases (many built opt & debug).
    - As mentioned before these releases are for simulation, central data reprocessing, cosmic rays, migration to new external software, etc., etc.
    - The development, production, and bug fix releases are copies of nightlies.
    - There are also several releases producing patches. These patch releases replace buggy code on the fly as the software starts up.
    - 44 servers (220 cores) are used to build the releases and do basic tests.
- It is our hope to get back to a simpler situation.
  - With our organization, we are managing the current situation well.

## Multiple Releases and Coordinators

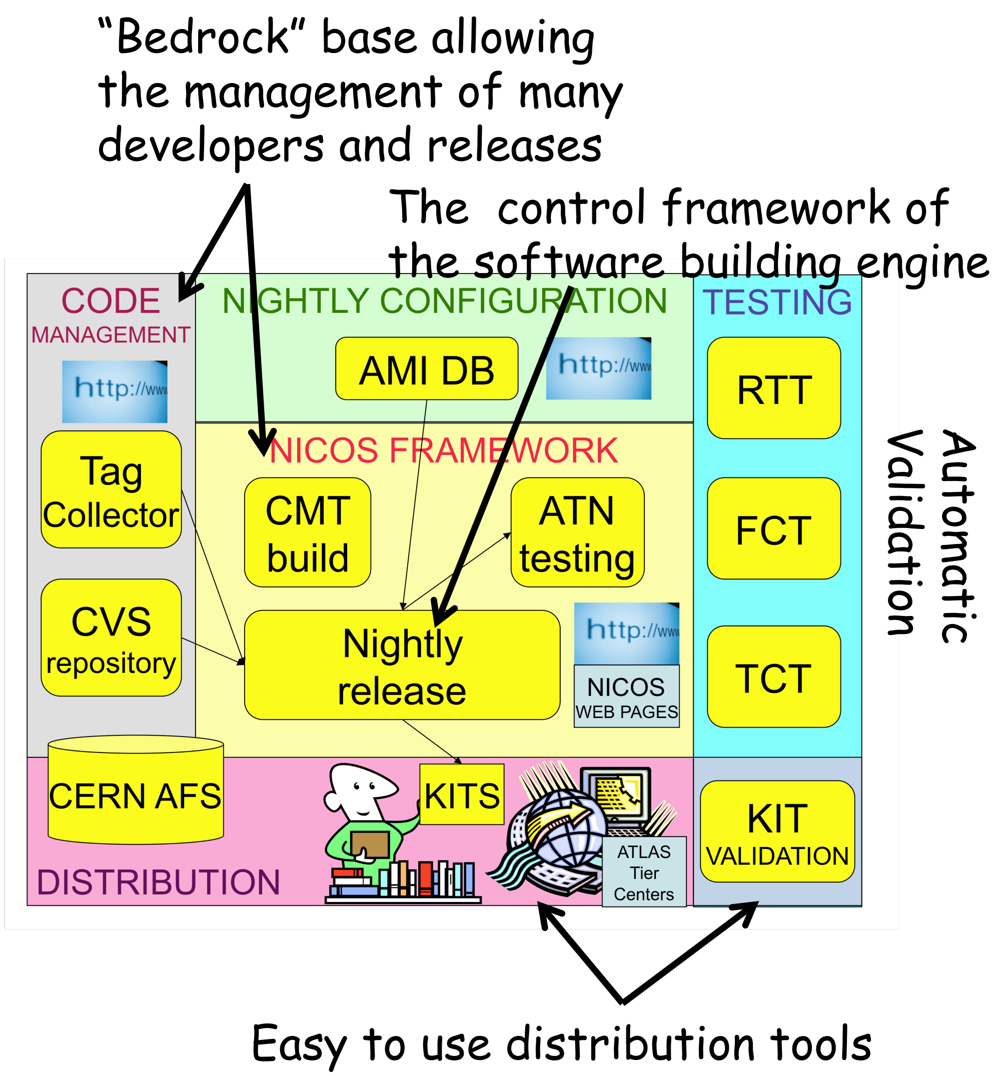
- The needs of the many software communities and our needs to validate new versions of external software (e.g. ROOT, GEANT) have lead to a intricate release coordination system to manage the multiple releases.
  - Each release has its own release coordinator who:
    - Controls what tags are allowed in the release
    - Checks the relevant validation tests each day
    - Moves new code from a "validation" (trial) state to a "development" (candidate) state to being in the final release.
    - Tracks problems along the way with Savannah tickets.
  - There are also two release builders on shift at all times to build the releases once the OK to build a release is given.
    - For important releases, many people may have to signoff while for minor releases, the release coordinator just asks the shifters to build it.
    - We hold weekly global release coordination meetings to track issues needing resolution before the pending releases can be built.
      - These meetings also look at the relative priority of the pending releases.

## Multiple Releases and Coordinators

- Even with aggressive release coordination and extensive validation, it takes time and multiple iterations to get a final production version of the software.
- Two mechanisms exist for adding adjustments and corrections to stable releases:
  1. Patch releases that allow the software to patch itself at runtime (does not require a full rebuild of the release).
  2. Bug fix releases that are full releases build rolling up the patches.
- It is a constant struggle to meet the release schedule.
  - This is our toughest issue.

# Software Release Tools

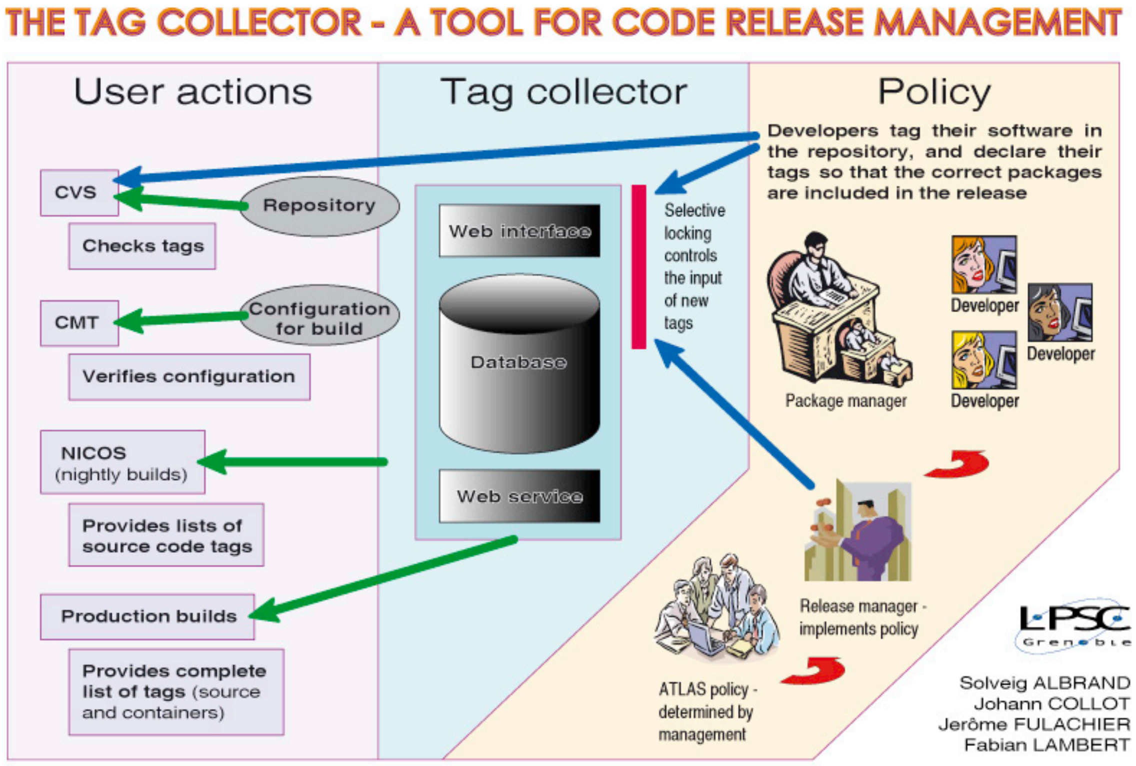
- The SIT uses a number of tools to build, test, and distribute the SW:
  1. CVS – the code repository that holds the code submitted by the developers
  2. Tag Collector (TC) – manages which software versions are used in the release
  3. CMT – manages software configuration, build, and use
  4. NICOS – drives nightly builds of the ATLAS software making use of a variety of tools
  5. Selected nightlies become official releases
  6. Four automatic systems validate the software builds
  7. CMT, CMT-based packaging tools, Pacman - create & install the software distribution kit.
- In addition to the above tools, we also use Doxygen to document our code automatically and 3 workbooks to teach developers/users how to use the software.



# Tag Collector (TC)

- The tag collector tool allows developers to select which code version (tags) in the CVS repository will be included in the release.
  - The TC is able to deal with the complicated ATLAS build environment and works well. It is still evolving to include new features. The TC is coded in Java and built on the AMI database.

- Recently added to TC:
  1. The ability to group a number tags into a "bundle" that must be approved jointly.
  2. Developers can ask a release coordinator to approve a tag.
  3. Developers can record whether tags passed validation tests.

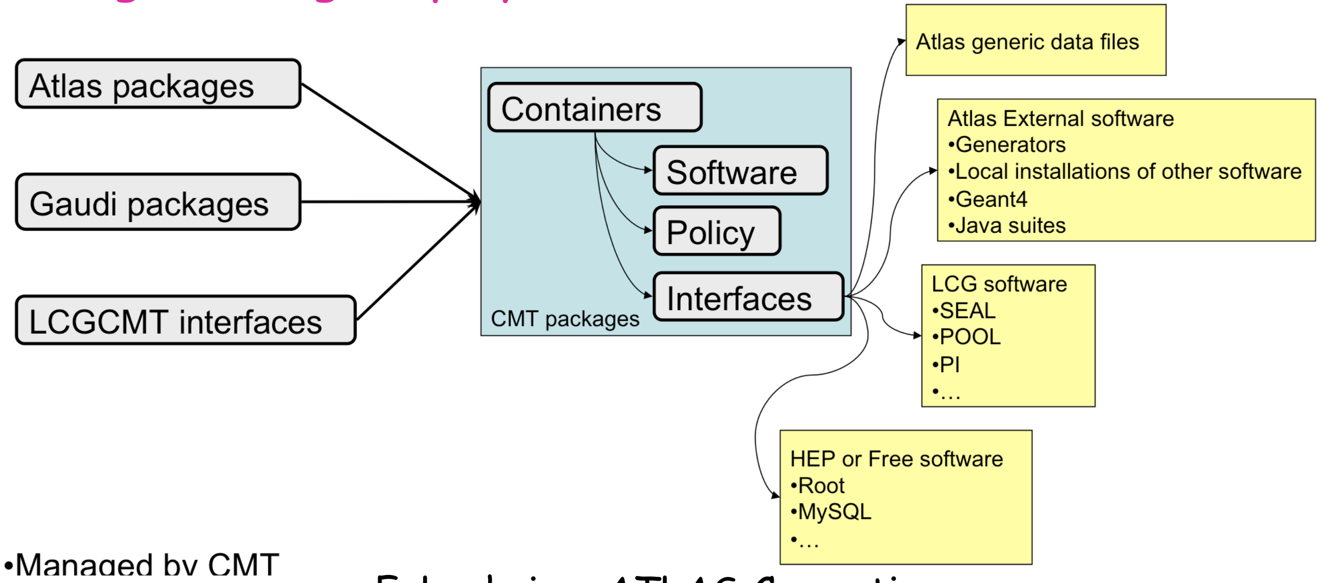

THE TAG COLLECTOR - A TOOL FOR CODE RELEASE MANAGEMENT
User actions
Tag collector
Policy
Developers tag their software in the repository, and declare their tags so that the correct packages are included in the release
CVS
Repository
Checks tags
CMT
Configuration for build
Verifies configuration
NICOS (nightly builds)
Provides lists of source code tags
Production builds
Provides complete list of tags (source and containers)
Web interface
Database
Web service
Selective locking controls the input of new tags
Package manager
Developer
Developer
Developer
Release manager - implements policy
ATLAS policy - determined by management
LPSC Grenoble
Solveig ALBRAND
Johann COLLOT
Jerôme FULACHIER
Fabian LAMBERT

## CMT

- CMT (Configuration Management Tool) is a configuration management tool based on management conventions that use a package-oriented principles
  - CMT is a C++ application available under UNIX/Linux, MacOSX, native Windows
- What CMT does:
  - Structures software packages: which project a package belongs to, name, version
  - Structures software projects: location, order in projects hierarchy, packages, strategies
  - Defines development work models: project management, development and integration work
  - Identifies the elements of configuration: projects, packages, applications, libraries, actions, context
  - Continues on next slide…

# CMT (2)

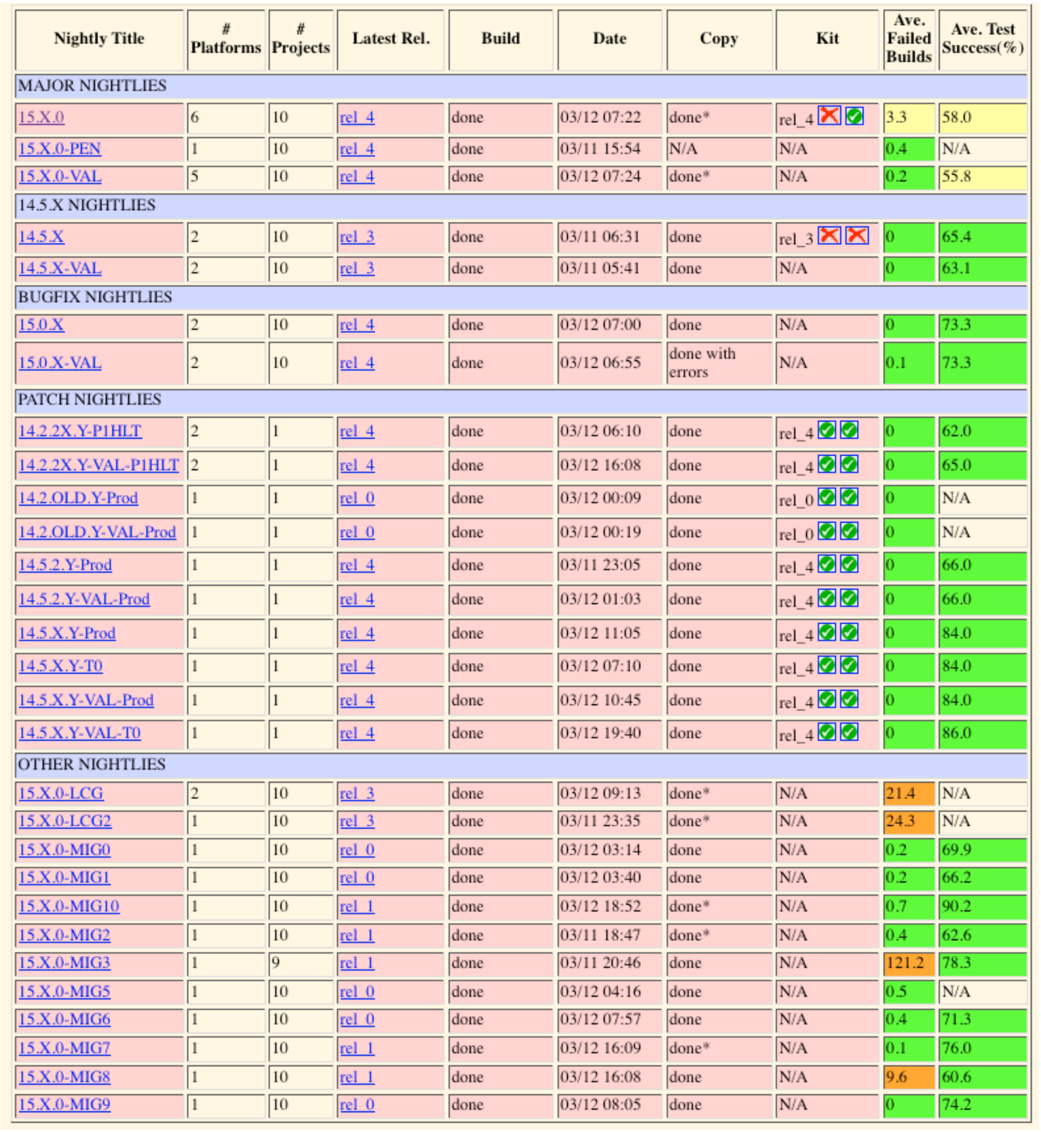
- What CMT does (continued):
  - Defines recurrent patterns in configuration management: where to install software, how to build shared libraries on many platforms, etc.
  - Queries the knowledge base to parameterize the development tools: cmt show macros, uses, ..., `cmt broadcast` [OPTIONS]... COMMAND
    - E.g., build the release with one command: cmt broadcast make
  - Identify and describe the activities of the software production (building, testing, deployment)



Computing in High Energy and Nuclear Physics
Prague | Czech Republic | 21 – 27 March 2009

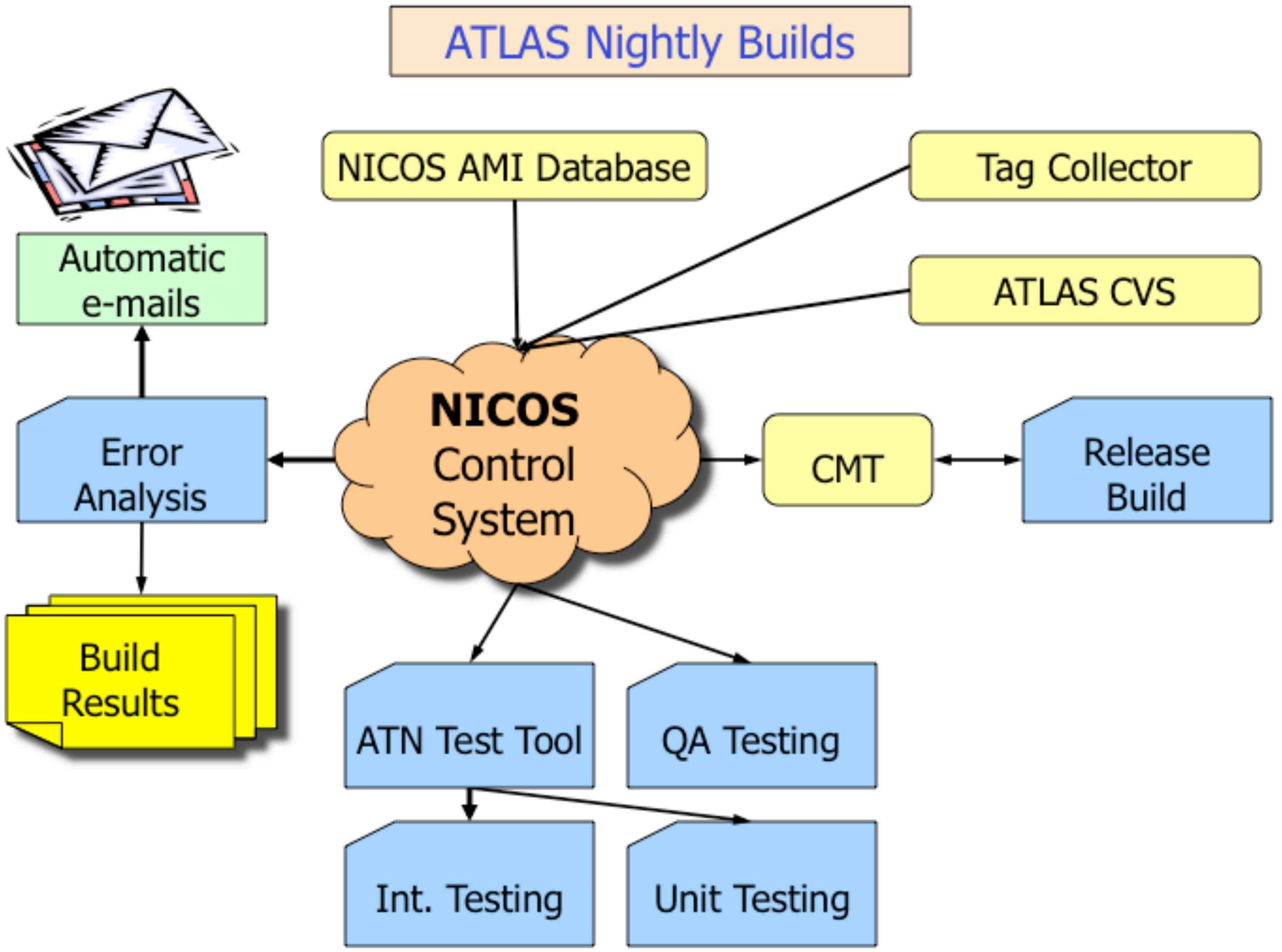
# NICOS

- The NICOS (NIghtly COntrol System) tool uses CMT to build many different software versions ("branches") each night (and some in the daytime too!).
  - As shown by the NICOS screenshot at right, ATLAS is making as many as 45 builds in each 24 hours.
- Lots of work has gone into making NICOS flexible.
  - NICOS uses the AMI database to store configurations.
    - AMI also underlies the tag collector.



| Nightly Title | # Platforms | # Projects | Latest Rel. | Build | Date | Copy | Kit | Ave. Failed Builds | Ave. Test Success(%) |
|---|---|---|---|---|---|---|---|---|---|
| MAJOR NIGHTLIES | | | | | | | | | |
| 15.X.0 | 6 | 10 | rel_4 | done | 03/12 07:22 | done* | rel_4 ✗ ✓ | 3.3 | 58.0 |
| 15.X.0-PEN | 1 | 10 | rel_4 | done | 03/11 15:54 | N/A | N/A | 0.4 | N/A |
| 15.X.0-VAL | 5 | 10 | rel_4 | done | 03/12 07:24 | done* | N/A | 0.2 | 55.8 |
| 14.5.X NIGHTLIES | | | | | | | | | |
| 14.5.X | 2 | 10 | rel_3 | done | 03/11 06:31 | done | rel_3 ✗ ✗ | 0 | 65.4 |
| 14.5.X-VAL | 2 | 10 | rel_3 | done | 03/11 05:41 | done | N/A | 0 | 63.1 |
| BUGFIX NIGHTLIES | | | | | | | | | |
| 15.0.X | 2 | 10 | rel_4 | done | 03/12 07:00 | done | N/A | 0 | 73.3 |
| 15.0.X-VAL | 2 | 10 | rel_4 | done | 03/12 06:55 | done with errors | N/A | 0.1 | 73.3 |
| PATCH NIGHTLIES | | | | | | | | | |
| 14.2.2X.Y-P1HLT | 2 | 1 | rel_4 | done | 03/12 06:10 | done | rel_4 ✓ ✓ | 0 | 62.0 |
| 14.2.2X.Y-VAL-P1HLT | 2 | 1 | rel_4 | done | 03/12 16:08 | done | rel_4 ✓ ✓ | 0 | 65.0 |
| 14.2.OLD.Y-Prod | 1 | 1 | rel_0 | done | 03/12 00:09 | done | rel_0 ✓ ✓ | 0 | N/A |
| 14.2.OLD.Y-VAL-Prod | 1 | 1 | rel_0 | done | 03/12 00:19 | done | rel_0 ✓ ✓ | 0 | N/A |
| 14.5.2.Y-Prod | 1 | 1 | rel_4 | done | 03/11 23:05 | done | rel_4 ✓ ✓ | 0 | 66.0 |
| 14.5.2.Y-VAL-Prod | 1 | 1 | rel_4 | done | 03/12 01:03 | done | rel_4 ✓ ✓ | 0 | 66.0 |
| 14.5.X.Y-Prod | 1 | 1 | rel_4 | done | 03/12 11:05 | done | rel_4 ✓ ✓ | 0 | 84.0 |
| 14.5.X.Y-T0 | 1 | 1 | rel_4 | done | 03/12 07:10 | done | rel_4 ✓ ✓ | 0 | 84.0 |
| 14.5.X.Y-VAL-Prod | 1 | 1 | rel_4 | done | 03/12 10:45 | done | rel_4 ✓ ✓ | 0 | 84.0 |
| 14.5.X.Y-VAL-T0 | 1 | 1 | rel_4 | done | 03/12 19:40 | done | rel_4 ✓ ✓ | 0 | 86.0 |
| OTHER NIGHTLIES | | | | | | | | | |
| 15.X.0-LCG | 2 | 10 | rel_3 | done | 03/12 09:13 | done* | N/A | 21.4 | N/A |
| 15.X.0-LCG2 | 1 | 10 | rel_3 | done | 03/11 23:35 | done* | N/A | 24.3 | N/A |
| 15.X.0-MIG0 | 1 | 10 | rel_0 | done | 03/12 03:14 | done | N/A | 0.2 | 69.9 |
| 15.X.0-MIG1 | 1 | 10 | rel_0 | done | 03/12 03:40 | done | N/A | 0.2 | 66.2 |
| 15.X.0-MIG10 | 1 | 10 | rel_1 | done | 03/12 18:52 | done* | N/A | 0.7 | 90.2 |
| 15.X.0-MIG2 | 1 | 10 | rel_1 | done | 03/11 18:47 | done* | N/A | 0.4 | 62.6 |
| 15.X.0-MIG3 | 1 | 9 | rel_1 | done | 03/11 20:46 | done | N/A | 121.2 | 78.3 |
| 15.X.0-MIG5 | 1 | 10 | rel_0 | done | 03/12 04:16 | done | N/A | 0.5 | N/A |
| 15.X.0-MIG6 | 1 | 10 | rel_0 | done | 03/12 07:57 | done | N/A | 0.4 | 71.3 |
| 15.X.0-MIG7 | 1 | 10 | rel_1 | done | 03/12 16:09 | done* | N/A | 0.1 | 76.0 |
| 15.X.0-MIG8 | 1 | 10 | rel_1 | done | 03/12 16:08 | done | N/A | 9.6 | 60.6 |
| 15.X.0-MIG9 | 1 | 10 | rel_0 | done | 03/12 08:05 | done | N/A | 0 | 74.2 |

## NICOS Architecture



## Validation

- ATLAS uses several tools to validate the building and the installation of its software.
  - AtNight (ATN) testing is integrated into NICOS runs over 300 "smoke" tests for most nightly branches during the nightly builds.
  - Full Chain Testing (FCT) checks the entire ATLAS software chain.
  - Run Time Tester (RTT) runs an extensive list of user defined tests against all of the software projects for 10 different builds each day.
  - Tier 0 Chain Testing (TCT) is used to check that the ATLAS software is installed and running correctly at the Tier 0.
  - Kit Validation (KV) is a standalone package that is used to validate that the ATLAS software is installed and functioning correctly on all ATLAS production sites (Tier 1/2/3).
- The release coordinators actively monitor each day's test results to see if new tag bundles can be promoted to candidate production releases and check that the candidate production release is working.

## Validation: Use of RTT

- The RTT system is a workhorse for validating ATLAS software.
  - The RTT system runs significant validation tests against 10 software builds each day.
  - The RTT runs between 10 & 500 test jobs against a release.
    - The test jobs can run for several hours and test a package intensively.
    - Currently the RTT runs ~1300 test jobs each day.
  - The RTT uses a pool of 12 cores to schedule the test jobs and 368 cores to run the test jobs.
    - Additional scheduling cores are currently being brought online.
    - The RTT also needs a small disk pool of 1.4 TB.
- The RTT system is highly scalable: if more tests are needed or more releases need testing, the RTT team can (and has!) added more nodes to the pool of servers used by the RTT system.

# CMT-Based Packaging Tools

- PackDist is a CMT package including a suite of scripts
- Use the CMT query mechanisms to visit the packages tree, retrieve the configuration/meta-data parameters and to generate the distribution kit as a set of packages in various packaging systems formats (E.g. Pacman)
- Packaging within the kit is done at the granularity of CMT projects (fast)
- Each packaged project is sub-divided into platform-dependent, platform-independent, source, and documentation parts (possible to install: selected platforms, source code)
- Typical usage during nightly builds
  - `> cmt make pack`
    CMT drives distribution kit build procedure (for a full release)
  - `> cmt make installation`
    CMT drives distribution kit installation process by installing the latest Pacman version and setting up the use of Pacman. to install the newly built kit.

## Distribution

- ATLAS has used Pacman to install the offline software for the last several years.
  - The installation has been done using pacman caches and mirrors.
- Recently we have been prototyping using "pacballs" to install the software. A pacball is:
  - Created using Pacman.
  - A self-installing executable.
  - Stamped with an md5sum in its file name.
- The pacballs come in two sizes:
  1. Large (~3 GB) for full releases (development, production, & bug fix).
  2. Small (50-200 MB) for the patch releases that are applied on top of a full release.
- Pacballs can be used to install the offline ATLAS software everywhere throughout the ATLAS collaboration from individual laptops to large production clusters. (The Tier 0 installation is different).

## Documentation

- ATLAS documents its code using Doxygen.
  - All developers are expected to put normal comments and special Doxygen comments into their source code.
    - Once a day the Doxygen processor formats the comments and header information about the classes and members into web pages.
  - Twiki pages are then used to organize the Doxygen-generated web pages into coherent content.
  - There are other ways to see the Doxygen pages including an alphabetical class listing and links from within the tag collector.
- ATLAS now has three "workbooks":
  1. The original introductory workbook.
  2. A software developers workbook.
  3. A physics analysis workbook.
- ATLAS has had two coding "stand-down" weeks to write documentation.
  - A third coding stand-down week is planned to happen soon.
- Significant time/effort has been spent to review the documentation.

## Final Thoughts

- The SIT has adapted our tools and methods to the growing demand to support more developers, releases, validation, and documentation.
  - Even though the SIT is small we provide excellent, effective central support for a very large collaborative software project.
- As ATLAS readies production code for data taking, producing working code on a schedule has been difficult but we have learned what is required to do the job:
  - Strong tools to ease the handling of millions of lines of code.
  - Lots of testing and code validation.
  - Lots of careful (human) coordination.
- We are continue to proactively look for way to further improve the ATLAS software infrastructure.

# Links

ATN: https://twiki.cern.ch/twiki/bin/view/Atlas/OfflineTestingInfrastructure#ATN

CMT: http://www.cmtsite.org/

Doxygen: https://twiki.cern.ch/twiki/bin/view/Atlas/DoxygenDocumentation

Pacman: http://physics.bu.edu/pacman/

NICOS: https://twiki.cern.ch/twiki/bin/view/Atlas/NIghtlyCOntrolSystem

RTT: https://twiki.cern.ch/twiki/bin/view/Atlas/OfflineTestingInfrastructure#RTT

SIT: https://twiki.cern.ch/twiki/bin/view/Atlas/SoftwareInfrastructureTeam

Tag Collector: https://twiki.cern.ch/twiki/bin/view/Atlas/TagCollectorInAtlas

Workbook: https://twiki.cern.ch/twiki/bin/view/Atlas/WorkBook