

OPEN ACCESS

Organization, management, and documentation of ATLAS offline software releases

To cite this article: S Albrand *et al* 2010 *J. Phys.: Conf. Ser.* **219** 042012

View the [article online](#) for updates and enhancements.

You may also like

- [Operation of the ATLAS trigger system in Run 2](#)
The ATLAS collaboration
- [The scientific potential and technological challenges of the High-Luminosity Large Hadron Collider program](#)
Oliver Brüning, Heather Gray, Katja Klein et al.
- [ATLAS: A High-cadence All-sky Survey System](#)
J. L. Tonry, L. Denneau, A. N. Heinze et al.



ECS
The
Electrochemical
Society
Advancing solid state &
electrochemical science & technology

DISCOVER
how sustainability
intersects with
electrochemistry & solid
state science research

Organization, Management, and Documentation of ATLAS Offline Software Releases

S Albrand¹, N Amram², K Black³, K Ciba⁴, A de Salvo⁵, J Fulachier¹, M Gallas
Torreira⁶, S Haywood⁷, V Jain⁸, I Kachaev⁹, F Lambert¹, S L Lloyd¹⁰, F
Luehring^{8,21}, E Moyse¹¹, E Obreshkov¹², A Pacheco Page^{6,13}, D Quarrie¹⁴, G
Rybkin¹⁵, P Sherwood¹⁶, B Simmons¹⁶, A S Thompson¹⁷, A Undrus¹⁸, H von der
Schmitt¹⁹, S Youssef²⁰, O Zenin⁹

¹LPSC/CNRS-IN2P3/UJF /INPG, 53 avenue des Martyrs, 38026 GRENOBLE
CEDEX France

²Ramat Aviv, Tel Aviv 69978, Israel

³Harvard University, 18 Hammond Street, Cambridge MA 02138 USA

⁴AGH-University of Science, (FPACS AGH-UST) al. Mickiewicza 30, PL - 30059
Cracow, Poland

⁵Università La Sapienza, Dipartimento di Fisica, Piazzale A. Moro, IT - 00185 Roma,
Italy

⁶CERN, CH - 1211 Geneva 23, Switzerland

⁷RAL, Harwell Science and Innovation Campus, Didcot, OX11 0QX, UK

⁸Indiana University, Bloomington IN 47405-7105, USA

⁹IHEP, Moscow Region, RU - 142 284 Protvino, Russia

¹⁰Queen Mary University of London, Mile End Road, London E1 4NS, UK

¹¹University of Massachusetts, 710 North Pleasant Street, Amherst, MA 01003, USA

¹²DESY, Hamburg and Zeuthen, Notkestr., D-22603 Hamburg, Germany

¹³IFAE, Edifici Cn, Universitat Autònoma de Barcelona, ES - 08193 Bellaterra
(Barcelona), Spain

¹⁴LBNL, MS50B-6222, 1 Cyclotron Road, Berkeley CA 94720, USA

¹⁵LAL, Univ. Paris-Sud, IN2P3/CNRS, Orsay, France

¹⁶University College, Gower Street, London WC1E 6BT, UK

¹⁷University of Glasgow, Glasgow G12 8QQ, UK

¹⁸BNL, Upton, NY 11973, USA

¹⁹Max-Planck-Institut für Physik, Föhringer Ring 6, 80805 München, Germany

²⁰Boston University, 590 Commonwealth Avenue, Boston MA 02215, USA

E-Mail: Fred.Luehring@cern.ch

Abstract. We update our CHEP06 [2] presentation on the ATLAS experiment software infrastructure used to build, validate, distribute, and document the ATLAS offline software. The ATLAS collaboration's computational resources and software developers are distributed around the globe in about 35 countries. The ATLAS offline code base is currently over 7

²¹ To whom any correspondence should be addressed.

million source lines of code in 10,000+ C++ classes organized into about 2,000 packages. More than 400 developers contribute code each month. Since our last report, we have developed a powerful, flexible system to request code versions to be included in software builds, made changes to our software building tools, increased the number of daily builds used to validate significant code changes, improved the tools for distributing the code to our computational sites around the world, and made many advancements in the tools to document the code.

1. Introduction

1.1. ATLAS

The ATLAS (A Toroidal LHC ApparatuS) **Error! Reference source not found.** project is one of the biggest collaborative efforts ever undertaken in the sciences: about 2,800 physicists and 1,000 students participate. The participants are drawn from more than 200 universities and laboratories in about 35 countries. The Large Hadronic Collider (LHC) provides beams of protons that collide and generate hundreds of subatomic particles 40 million times per second. The LHC is a facility at CERN, the European Organization for Nuclear Research, located near Geneva, Switzerland. The large scale of the ATLAS offline software is driven by the complexity and scope of the ATLAS detector.

1.2. Collaborative Nature of the ATLAS Offline Software

The offline software effort is a collaboration of physicists, students, and computing professionals drawn from throughout the ATLAS collaboration. Currently ATLAS has over 800 people registered as offline software developers with approximately 400 developers submitting new or revised code each month. The ATLAS offline code base has approximately 7 million lines of code organized in approximately 2,000 packages. The main code languages used are C++ and Python with significant amounts of code written in Fortran/Fortran 90, SQL, PERL, and Java. Since May of 2000 there have been 15 major production releases of the ATLAS offline software and hundreds of development releases. It is the responsibility of the Software Infrastructure Team (SIT) to maintain this code base and build the ATLAS offline software (Athena). Note that while the ATLAS shares much of its code between the online and offline code efforts this paper describes work on the offline software.

2. The Software Infrastructure Team

2.1. Membership of the SIT

The SIT is made up of approximately 30 people: physicists, computing physicists, students, and computing professionals. While there is a core of four people who are dedicated to (and paid by) the ATLAS project, most SIT members are volunteers who spend a relatively small fraction of their time working on SIT-related activities. We estimate that the total effort on the SIT is equivalent to about 12 FTEs. Mirroring the ATLAS software developer community, the SIT members are spread throughout Europe, Israel, and the US with the dedicated personnel being at CERN. The part-time personnel and their institutes generally receive credit from the ATLAS collaboration for doing this service work.

2.2. Organization of the SIT

The SIT has a simple organization with a single physicist who serves as convener and organizes the effort. In addition one of the ATLAS software project leaders serves on the SIT and looks after the portions of the work based at CERN. The SIT has face-to-face meetings approximately quarterly at ATLAS software workshops and biweekly phone conferences in addition to hundreds of e-mail messages each week. Frequently the project leader and convener work to find additional people to cover new tasks that arise over time. In general this works very well and tasks get successfully

completed but the SIT activity is always a little short of having the full number of people needed to do everything required. The SIT activity is generally organized around the ATLAS offline software release schedule.

3. ATLAS Offline Software Releases

3.1. Software Organization

In addition to the large number of developers and the large amount of code, the SIT is supporting several versions (releases) of the software because of requests from different communities within ATLAS. Many different ATLAS communities use portions of the offline software for their own needs and want to modify “their” part of the code without being affected by changes made by other groups. For example, a group working on reprocessing cosmic ray data, may not want changes that trigger community wants and *vice versa*. Having releases for different groups puts a large stress on the SIT because often priorities conflict between the various groups. In addition, new versions of external packages (e.g. ROOT, GEANT) need to be included in the software from time to time. To accommodate the needs of the developer community, the SIT builds many different versions of the ATLAS software each day.

Each software version is organized into about 10 software projects and these projects in turn contain about 2,000 software packages. A package is always maintained in a single project and organization of the project and packages serves to both define and provide a logical structure for the software dependencies. Examples of projects are core software, simulation, and reconstruction. The projects are organized to reflect the order of the ATLAS offline data processing chain. Figure 1 shows the project dependencies for the ATLAS offline and online software projects. There are also container packages that group packages so that they correspond to parts of the ATLAS detector (Inner Detector, Calorimeter, etc.) and have different hierarchy then the software projects. A package is also in a single container but the structure from these container packages spans multiple projects (e.g. the Inner Detector has packages in many projects: simulation, reconstruction etc.)

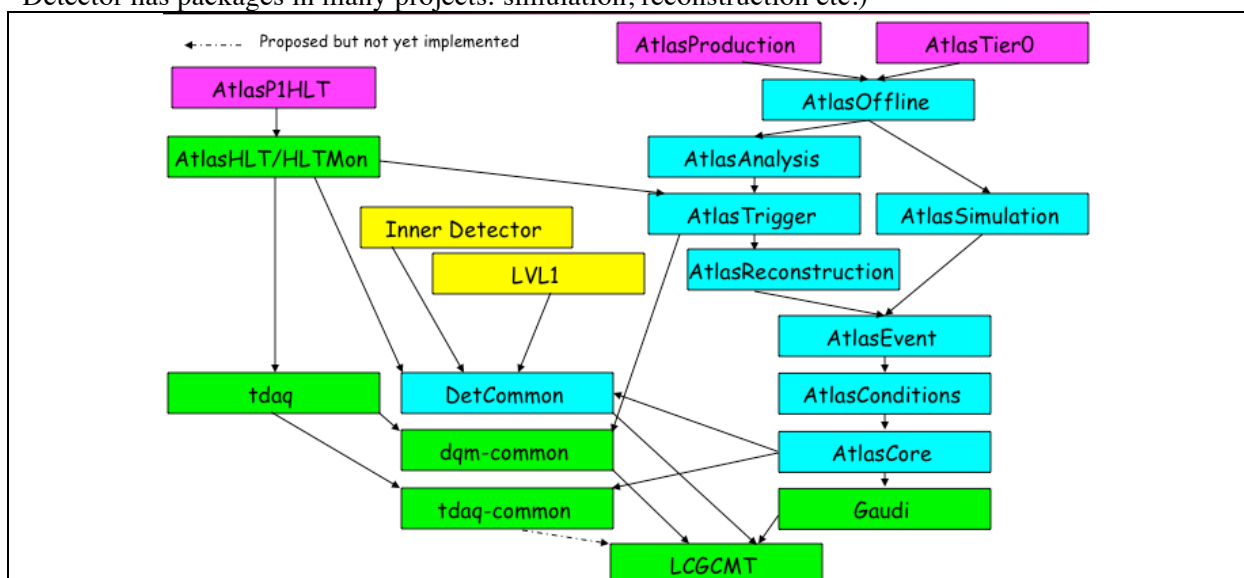


Figure 1 The project dependencies for the ATLAS offline and online software. The SIT is responsible for the projects between AtlasCore and AtlasProduction / AtlasTier0.

3.2. Release Strategy

The SIT has moved to a complex strategy for releasing many versions of the offline code developed in parallel. Originally [1], our scheme consisted of builds each night, a series of development releases each month, a candidate production release each six months, and then a rapid series of bug-fixing

releases based on the candidate release until production-quality code was achieved. Under this scheme, we would at times have two branches active: the bug-fixing branch for a particular release that was accepting only fixes to known bugs in the candidate production release, and the next development release that was open to allow development of new features in the code. However it was realized fairly quickly that this scheme was inadequate for large collaboration like ATLAS where there is a need to have a usable version of the nearly latest code each day to develop against and where many development efforts are being made in parallel[3].





















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-PIHLT	2	1	rel_4	done	03/12 06:10	done	rel_4  	0	62.0
14.2.2X.Y-VAL-PIHLT	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

Figure 2 Display from the NICOS system showing the results of one day's builds and ATN tests.

Therefore additional nightly releases were added. Firstly a validation (trial) nightly was added for testing proposed new code before the new code could be put into the development (candidate) nightly. This allowed the development nightly to be usable most mornings for test and development while at the same time providing a way to test new code. Secondly daily builds were added for such things as the 64 bit platform, the SLC5 operating system, and the gcc 4.3.2 compiler (currently, our production builds are on SLC4/32 bits/gcc 3.4.6). Thirdly many nightly builds designated as *migration* builds were added for testing larger, invasive changes to the code for specific purposes. These additional migration nightly releases were added to allow several developer communities to work on potentially disruptive migrations without adversely impacting the primary nightlies. As their name suggests the intention is that such migrations eventually be merged into the primary nightly releases. This added flexibility to the release building strategy at the expense of occasionally needing to expend a significant amount of work to combine the various parallel code versions into a single, fully-functional

version of the offline code. Figure 2 shows one day's builds and illustrates the large number of builds. An automatic tag "sweeping" process is invoked from time to time to copy new package versions from the various builds into a production candidate release. The object of tag sweeping is to keep the various software versions from diverging. In addition to release builds, a scheme was added to allow for a reasonable number of runtime patches to be applied to the code. The runtime patches can change source code (C++, Python, etc.) but not header files that would require recompilation of client packages.

Currently ATLAS is building about 30 releases each night with most releases having both an optimized and a debug version built which results in a total of about 45 builds. ATLAS is currently using 44 servers containing a total of 220 CPU cores to build the releases. In spite of the complexity of the current scheme we are handling the release building well.

3.3. Multiple Releases and Coordinators

Our way of managing multiple releases is to have multiple release coordinators: one per open release. A release coordinator is responsible for planning the goals of a release, deciding what new code versions ("new tags") are allowed into the release, validating that the newly submitted code works properly, and checking that the release is capable of providing the desired functionality. Currently as ATLAS scales up for LHC running there are typically four or five releases open (not counting migration builds) each with its own release coordinator. In particular, the release coordinators are responsible for accepting submitted code from the validation release to the development release. ATLAS also has two release building shifters available at most times, so that the release coordinators can ask for an official build of their release even on weekends and evenings. For important releases (i.e. those that will be used for major productions), the software project coordinators, validation managers, chief architect, and other responsible representatives must sign off before a release is built. Weekly Global Release Coordination (GRC) meetings attended by representatives of all concerned parties are held. In addition to looking at the status of the progress toward the release, the GRC meeting also looks at the relative priority of the various pending releases.

Even with aggressive release coordination and extensive validation, it takes time and multiple iterations to get a final production version of the software release. Two mechanisms exist for adding adjustments and corrections to stable releases:

1. Patch releases that allow the software to patch itself at runtime (this does not require a full rebuild of the release). For a given candidate release, patch releases are issued at two week intervals with each patch release being a superset of the previous releases.
2. Bug-fixing releases that are full releases built using all of the fixes in the most recent patch release. This results in faster startup of the offline software by compiling in the patches during building rather than applying them on the fly at the start of a job.

Even with these two patching mechanisms, it is a constant struggle to meet the release schedule. Meeting the schedules is the SIT's toughest job.

4. ATLAS Software Release Tools

The SIT uses a number of tools to build, test, and distribute the SW (see Figure 3):

1. **CVS** – the code repository that holds the code submitted by the developers [4].
2. **Tag Collector** – manages which software packages and versions are used in the release [5].
3. **CMT** – manages software configuration, build, and use [6].
4. **NICOS** – drives nightly builds of the ATLAS software [7].
5. **Scripts** – convert selected nightly builds into official releases.
6. **Validation Tools** – five systems validate the software builds [9][10][11][10].
7. **Packaging & Installation Tools** – create & install the software distribution kit [12][12].

In addition to the above tools, we also use **Doxygen** [14] to document our code automatically, a **TWiki** [15] for written documentation, and three **Workbooks** [16] to teach developers/users how to use the software.

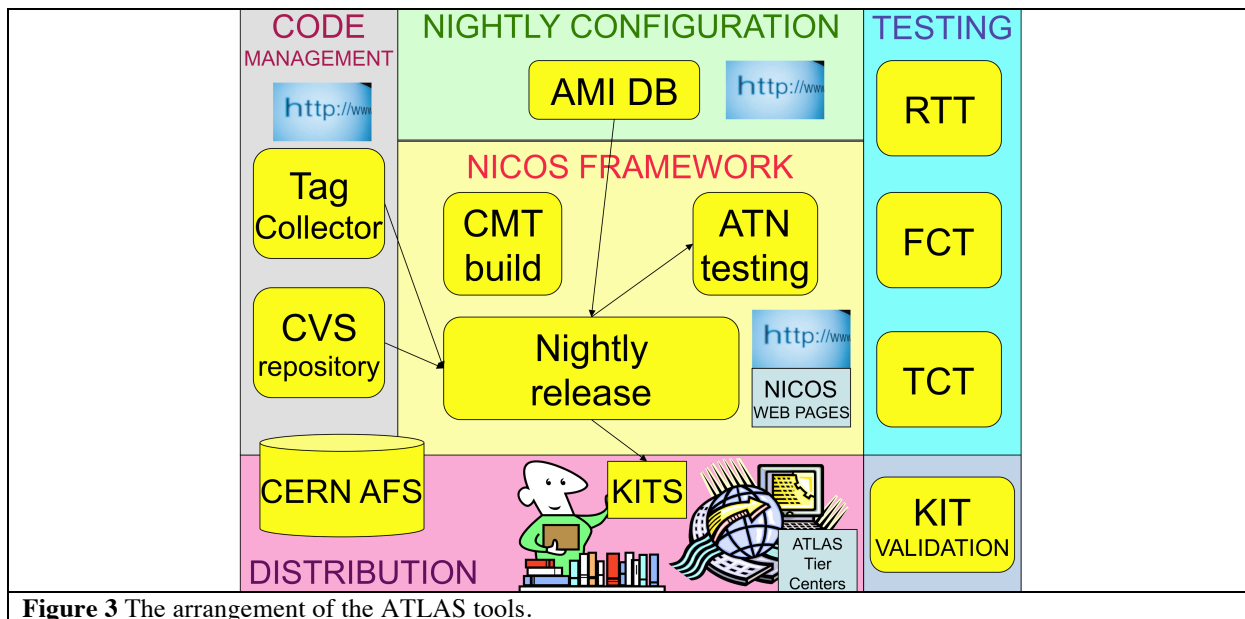


Figure 3 The arrangement of the ATLAS tools.

4.1. Software Building Tools

4.1.1. Tag Collector

The Tag Collector (TC) [5] tool allows developers to select which code version (tags) in the CVS repository will be included in the release. To be clear, a “tag” is a user-named version of a code package. The tagging of code versions is done within CVS. The TC has been improved to support the complicated ATLAS build environment. The TC is working well and is still being evolved to include new features.

Since our report at CHEP06 [2], the following features have been added to TC:

- The ability to group a number tags into a “bundle” that must be approved or rejected jointly. This new feature deals with dependencies between new tags and saves lots of error-prone and laborious manipulation of the list of selected tags when a change requires updates to more than one software package.
- Developers can ask a release coordinator to approve a tag or bundle for inclusion in the development version nightly (i.e. accept that a tag has been successfully validated).
- Developers can record whether tags passed validation tests after checking the tests. This saves the release coordinators from having to check the test of every package in a bundle and provides a record of when and who checked the validation tests.

The TC is coded in Java and built on the ATLAS Metadata Interface (AMI) application[17].

4.1.2. Configuration Management Tool

The Configuration Management Tool (CMT) [6] is based on management conventions that use package-oriented principles. CMT does the following:

- Structures software packages: which project a package belongs to, name, and version.
- Structures software projects: location, order in projects hierarchy, packages, and strategies.
- Defines development work models: project management, development and integration work.
- Identifies the elements of the build configuration: projects, packages, applications, libraries, actions, and context.
- 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 using this syntax:
`> cmt show macros, uses`

> `cmt broadcast [OPTIONS]... COMMAND`

- E.g., build the release with one command: `cmt broadcast make`
- Identifies and describes the activities of the software production (building, testing, deployment).

The way that users, developers, and release builders interact with CMT is through text files called `requirements` and `projects` files. These files provide a description of the package and project dependencies that allow CMT to construct appropriate `makefiles` to build the release. Each CMT software package contains a `requirements` file in a subdirectory called `cmt/`. Using these `requirements` files CMT is able to build the entire release in the correct order so that packages that depend on other packages are compiled in sequence. ATLAS has developed a tool called *checkreq* [17] that checks that the contents of the requirement files match the include statements in the source code and also checks for various syntactical errors in the requirements files.

4.1.3. Nightly Control System

The Nightly Control System (NICOS) tool [7] uses CMT to build many different versions (“branches”) of the ATLAS offline software each night (and some in day too!). NICOS also runs user-defined validation tests on the newly built software (see details in the next section). At the time of this report, ATLAS is making about 45 builds every 24 hours and significant work has gone into scaling up NICOS to handle the large number of builds, as well as into making NICOS flexible; a recent major improvement NICOS uses the AMI database to store the configuration of each nightly build. The web pages that NICOS uses to present the build results and ATN [8] test results (see section 4.2.1.) have been improved to make it easier for the release coordinators and developers to spot problems and investigate them. Considerable thought has gone into the usability of these web pages. Since AMI also underlies the Tag Collector, integration between NICOS and the Tag Collector is improved.

4.2. Software Validation Tools

4.2.1. Introduction

ATLAS uses several tools to validate the building and the installation of its software:

- AtNight (ATN) testing [8] is integrated into NICOS and runs over 300 “smoke” tests for most branches during the nightly builds. ATN can test a single routine or the functionality of simulation, reconstruction, or trigger on a few events.
- Full Chain Testing (FCT) [8] checks the entire ATLAS offline software chain (event generation, simulation, digitization, reconstruction, and analysis) and is run once a day.
- Tier-0 Chain Testing (TCT) is used to check that the ATLAS software is installed and running correctly at the Tier-0. This testing is very similar to the FCT except it applies to the software installed the CERN Tier-0 site where the software release is updated much more frequently than the production offline release run on the Grid.
- Run Time Tester (RTT) [9] runs an extensive list of user defined tests against all of the software projects for 10 different builds each day. Currently the RTT system will only test the packages in one project but it is being upgraded to incorporate the full-chain testing functionality of the FCT and TCT. The RTT tests can be quite lengthy and run for several hours. Further details are in the next section.
- Kit Validation (KV) [10] is a standalone package that is used to validate that the production version of the ATLAS software is installed and functioning correctly on all ATLAS Grid 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.

4.2.2. The RTT Software Validation System

The Run Time Tester (RTT) system [10] is the most intensive validation tool for ATLAS offline software. The RTT system runs a number of validation tests against 10 software builds each day with each build undergoing between 10 and 500 test jobs. The user defines each test with XML statements and supplies the necessary data files. The RTT test jobs can run for several hours. It is foreseen that some RTT tests could run for several days but currently this is not the case. Recently the RTT has been running ~1300 test jobs each day. The RTT system is written in Python and uses a dedicated cluster to run the validation tests. The cluster has a pool of 13 servers 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 of AFS-mounted disks. 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. The results of the tests are available on a website and log and output files can easily be downloaded for further examination.

4.3. Software Distribution Tools

The ATLAS distribution kit is designed to install and manage the offline ATLAS software as well as the required external software. It is used at the ATLAS experiment site, ATLAS trigger computing clusters, on AFS at CERN, all collaborating institutes, and also beyond the collaboration at all the Grid sites providing resources to ATLAS. The distribution kit is built with the CMT-based packaging tools. The distribution kit is installed for offline use with the *Pacman* installation tool.

4.3.1. CMT-Based Distribution Kit Packaging Tools

The ATLAS software distribution kit packaging system, PackDist, is a CMT package including a suite of scripts [12]. The system uses the CMT query mechanisms to visit the package tree, to 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 for faster building and installation. Each packaged project is sub-divided into platform-dependent, platform-independent, source, and documentation parts. The needed external software packages are also packaged and the ATLAS project packages are made dependent on them. It is possible to install selected platforms and/or the source code. The SIT typically provides the kit for a core set of widely used platforms. With minor modifications made by users, the kit installations work on many additional platforms beyond the platforms officially supported by the SIT. The distribution kit is built for every official full or patch release and for the majority of nightly releases.

Typical CMT usage during nightly builds:

- > `cmt make run` 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.

4.3.2. Pacman

ATLAS has used Pacman [13] to install the offline software for the past several years. The installation has been done using Pacman caches, mirrors, and snapshots. Recently we have been prototyping using “pacballs” to install the software. A pacball is a self-installing executable created using Pacman. It includes the ATLAS distribution kit for a particular release and a Pacman version. The pacball has an associated *md5sum* that is recorded in the pacball filename. Calculating the md5sum on a downloaded pacball allows a site downloading it to check that the pacball is not corrupted. ATLAS pacballs come in two sizes: Large (~3 GB) for full releases (development, production, and bug-fixing) and small (50-200 MB) for the patch releases that are applied on top of a full release.

The ATLAS pacballs are designed to install the offline ATLAS software anywhere from individual laptops to large production clusters. The pacballs are distributed using both a web server and the ATLAS DQ2 data distribution system. After being created at the ATLAS CERN-based Tier-0, all pacballs are propagated to all ATLAS Tier-1 sites. The Tier-0 site has two copies of the ATLAS

installation: one that is created during the building of ATLAS release and one that done in the same way using pacballs that all other sites install the software.

4.4. Documentation Tools

4.4.1. Doxygen

ATLAS documents its code using Doxygen [14] and all ATLAS developers are expected to put normal comments and special Doxygen comments into their source code. Doxygen comments work with C, C++, and Python code and provide a cross-referenced set of class names and member function names. The Doxygen processor runs once a day to format the comments and header information about the classes and members into web pages. User-written TWiki pages are 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 to the Doxygen content for each package from within the Tag Collector. In an effort to increase the amount of code with Doxygen content, ATLAS has had two coding “stand-down” weeks to write documentation and a third coding stand-down week is planned. In addition, significant time and effort has been spent to reviewing the Doxygen documentation and the supporting TWiki pages.

4.4.2. TWiki

ATLAS uses a TWiki [15] to generate most user-provided content and we have moved most computing-related static web pages to the TWiki. The decision to use the TWiki was made because wikis have proven to be easy to use and flexible. The TWiki now contains almost all SIT web content. After significant work by the CERN computer centre staff the ATLAS TWiki was split into two areas: one that is publically available and one that is only available to members of the collaboration. The SIT is still dealing with how to search for information on the private TWiki because the private TWiki is no longer visible to search engines such as Google. Additional details on the use of TWiki by ATLAS can be found in the proceedings article for the ATLAS TWiki poster shown at CHEP09[15].

4.4.3. Workbooks

ATLAS adopted the excellent concept of the “workbook” [16] from the BaBar experiment at SLAC and the OPAL experiment at CERN. A workbook is a written set of annotated examples of how to do certain tasks with the software. New users following the workbook quickly learn how to do basic tasks with the ATLAS software. ATLAS now has three “workbooks”:

1. The original introductory workbook on how to get started with the offline software.
2. A software developer’s workbook for learning how to develop software packages.
3. A physics analysis workbook that introduces the tools for analyzing ATLAS data.

The reception of the workbooks by the ATLAS user and developer communities has been very favorable. The CMS experiment has adopted the workbook system based on ATLAS’s success with its workbooks.

5. Conclusion

The SIT has adapted its tools and methods to the ever-increasing demand by the ATLAS collaboration to support more and more developers, releases, validation, and documentation. Even though the SIT has small number of FTEs, it provides 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:

- Lots of sophisticated tools to ease the handling of the large code base.
- Lots of care testing and validating the code to ensure that it functions properly.
- Lots of careful (human) coordination to organize the effort and keep track of the details.

We continue to proactively look for way to further improve the ATLAS software infrastructure.

References

- [1] The SIT Collaboration, Albrand, S et al. 2006 Organization and management of ATLAS software releases Proceedings of CHEP 2006 Mumbai India
 Obreshkov E et al. 2008 Organization and management of ATLAS software releases *Nucl. Instrum. Meth.* **A584** 244-251
- [2] The ATLAS Collaboration, Aad G et al. 2008 The ATLAS Experiment at the CERN Large Hadron Collider, *JINST* **3** S08003
- [3] <https://twiki.cern.ch/twiki/bin/view/Atlas/AtlasReleaseValidation>
- [4] <http://www.nongnu.org/cvs/>
- [5] Albrand S, Collot J, Fulachier J and Lambert F 2004 The tag collector – a tool for ATLAS Code Release Management Proceedings of CHEP 2004 Interlaken Switzerland
- [6] Arnault C 2000 Configuraton Management Tool Proceedings of CHEP 2000 Padova Italy
 Arnault C 2001 Experiencing CMT in software production of large and complex projects Proceedings of CHEP 2001 Beijing China
<http://www.cmts.site.org/>
- [7] Undrus A 2003 Proc. Int. Conf. on Computing in High Energy and Nuclear Physics CHEP'03 (La Jolla, USA) e-Print hep-ex/0305087 pp TUJT006
<http://www.usatlas.bnl.gov/computing/software/nicos/index.html>
- [8] Undrus A 2004 Proc. Int. Conf. on Computing in High Energy and Nuclear Physics CHEP'04 (Interlaken, Switzerland) CERN 2005-002 pp 521-3
- [9] Zenz S 2008 A testing framework for the ATLAS offline simulation and reconstruction chain ATL-SOFT-INT-2009-001; ATL-COM-SOFT-2008-011 Geneva CERN
- [10] Ciba K, Richards A, Sherwood W and Simmons B 2009 The ATLAS RunTimeTester software To be published in the proceedings of CHEP 2009 Prague Czech Republic
- [11] Brasolin F and de Salvo A 2009 Benchmarking the ATLAS software though the Kit Validation engine To be published in the proceedings of CHEP 2009 Prague Czech Republic
- [12] Arnault C, de, Salvo A., George S and Rybkine G 2004 Proceedings of CHEP 2004 Interlaken Switzerland
- [13] <http://physics.bu.edu/pacman>
- [14] <http://www.Doxygen.org/>
<https://twiki.cern.ch/twiki/bin/view/Atlas/DoxygenDocumentation>
- [15] Amram N, Antonelli S, Haywood S, Lloyd S, Luehring F and Poulard G 2009 The use of the TWiki web in ATLAS To be published in the proceedings of CHEP 2009 Prague Czech Republic
- [16] <https://twiki.cern.ch/twiki/bin/view/Atlas/WorkBook>
- [17] <http://ami.in2p3.fr/>
- [18] <https://twiki.cern.ch/twiki/bin/view/Atlas/SoftwareDevelopmentWorkBookCheckReq>