Automated Tests in NICOS Nightly Control System

A. Undrus*, Brookhaven National Laboratory, Upton, NY 11973, USA

Abstract

Software testing is a difficult, time-consuming process that requires technical sophistication and proper planning. This is especially true for the large-scale software projects of High Energy Physics where constant modifications and enhancements are typical. The automated nightly testing is the important component of NICOS, NIghtly COntrol System, that manages the multi-platform nightly builds based on the recent versions of software packages. It facilitates collective work in collaborative environment and provides four benefits to developers: repeatability (tests can be executed more than once), accumulation (results are stored and reflected on NICOS web pages), feedback (automatic e-mail notifications about test failures), user friendly setup (configuration parameters can be encrypted in the body of test scripts). The modular structure of NICOS allows plugging in other validation and organization tools. NICOS supports tests of different granularity level and purpose. The low level structural tests reveal inconsistencies in package configuration and bugs localized in components and interfaces. The results for these tests are published for each package of the software project. The integration tests find bugs at levels of users scenarios and NICOS generates the special web page with their results. The NICOS tool is currently used to coordinate the efforts of more than 100 developers for the ATLAS project at CERN.

INTRODUCTION

The software products for High Energy Physics are often responsible for generation, selection, and analysis of large datasets. If left unchecked, errors can easily propagate through all stages of a research cycle. Thoroughly organized testing requires substantial resources but allows to avoid more costly repairs in data production. The automated nightly builds and tests become a major component in the collaborative organization of High Energy Physics projects. The extensive nightly testing allows to prepare the software releases of high quality that can be promptly validated and applied for research tasks.

NICOS [1] is a versatile nightly build tool that was originally created for the software development in ATLAS [2] experiment. It provides a framework for code development and tests of different scopes and types. NICOS operates on UNIX-like platforms and can work with external release management and test tools, such as CMT [3], QMTest [4]. The NICOS tool is included in the tool library of the LHC Computing Grid Project [5].

* undrus@bnl.gov

NICOS DESIGN

NICOS [1] is the nightly build system that

- provides options for version management and number of releases in a cycle,
- builds the software releases,
- performs testing,
- informs programmers about results with dynamic web pages and personal e-mails.

It is a portable PERL based tool with the modular organization shown in Fig. 1. The modules are able to serve as interfaces to external tools such as code repositories, test and release management tools. NICOS automatically posts the information about the progress of nightly builds, identifies compilation problems, determines the outcome of tests, and creates the web pages with build and test results. For faster feedback the e-mail notifications about compilation and test problems can be automatically distributed to responsible developers.

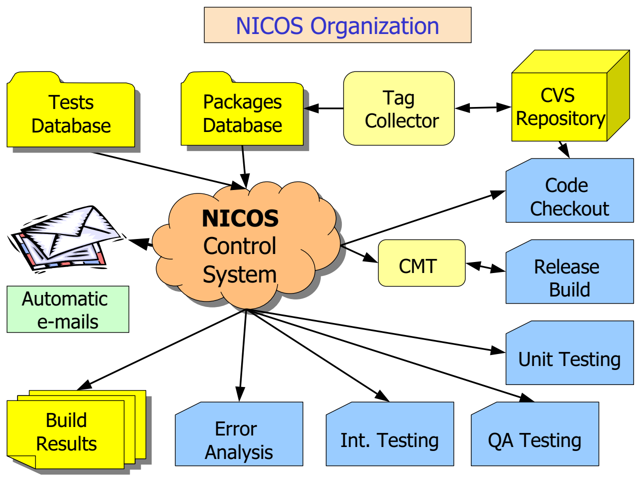

Figure 1: NICOS modular organization.

NICOS is easy to use for both administrators and software developers. The NICOS project configuration is stored in a single XML file named `nicos_cache`. The steps of building process is associated with its markup tag. Every tag consist of a tag name, sometimes followed by an optional list of tag attribute. The markup tag is followed by commands, including environment definitions, needed to be executed at the step. The versions of packages for the nightly builds can be specified in the NICOS packages database file or supplied by an external tool. Another NICOS database stores information about tests names and locations. NICOS databases are text files that can be easily modified.

ATLAS software releases comprise about 1000 packages. The following external tools are used for their management and interfaced to NICOS:

- ATLAS Tag Collector [6] is the web interfaced database application. Developers are able to interactively select the tags from ATLAS CVS repository for the nightly releases. NICOS retrieves the list of packages versions from the Tag Collector database before the build starts.

- CMT [3] is the configuration management tool that defines the conventions for structuring software releases and describes the package properties, constituents, and dependencies. In particular, it defines the build and run-time environment for ATLAS software releases.

In addition, about 10% of the ATLAS offline software code is generated automatically through NICOS integration with the NOVA database [7] storing 30K parameters for the detector description. NICOS supports the automatic code generation of more than 300 NOVA objects classes for a persistent data access in the simultaneous parallel software builds and automatic nightly tests of the NOVA database update procedures.

TESTING FRAMEWORK

NICOS Control Tool provides a framework for tests of different manners and levels of focus as shown on Fig. 2. In NICOS organization the separate modules are responsible for running tests of different types. In this section the NICOS capabilities are illustrated by the concrete examples from the organization of ATLAS nightly builds.

- Although the verification of compilation results is not a test, the early detection of defects in compilation and linking is important for the decision on further testing. NICOS checks for compilation errors and publishes results immediately after completion of `make` and before the testing begins.

- Quality Assurance (QA) tests are usually performed without the running the system. They evaluate the code design, organization, and documentation. In ATLAS nightly builds the code checking tool verifies the description of package properties in CMT `requirements` files, in particular the consistency of packages dependencies.

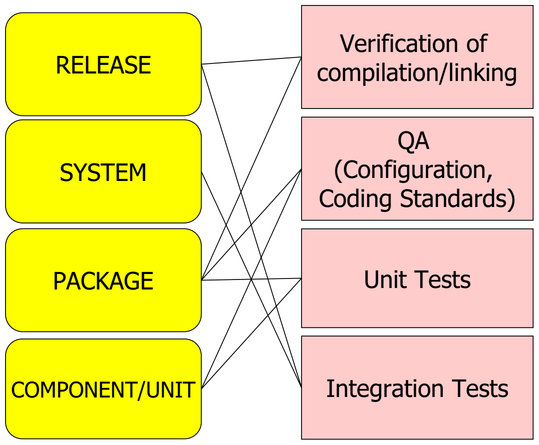
RELEASE
SYSTEM
PACKAGE
COMPONENT/UNIT
Verification of compilation/linking
QA (Configuration, Coding Standards)
Unit Tests
Integration Tests

Figure 2: The types and scopes of tests. The tests of one type can vary in focus.

- Unit tests are often based on the structural properties of the source code and applied to the individual software components (such as C++ classes). The helper tool for unit testing of C++ programs, CppUNIT [8] is integrated in CMT environment of ATLAS releases. This tool provides the common test driver for all test cases. The driver allows to run tests automatically and get the summary. Unit tests can sometimes be focused on the properties of the components collections, such as the packages in a software release. In ATLAS control framework ATHENA [9] software uses the provided common services (job configuration, data store, message streams etc.). Therefore the tests focused on package functionalities require the operational core of the framework. NICOS allows users to decide how to classify such tests (unit or integration). The results of QA assurance and unit tests are summarized for each package of a release.

- Integration tests show that the major systems of the software release work well and communicate with other systems successfully. NICOS supports the basic functionalities for integration testing: organization tests in suites, search for test scripts in specified areas of the release, preliminary evaluation of results, and publishing the separate web page summary. In case of large software systems the external tools can be plugged in for extension of these functionalities. In ATLAS nightly builds QMTest [4] tool is used for ATLAS integration tests organization and validation. It supports the regression evaluation that compare the output of the current test with previous (or known) values. The tests are arranged in suites. The suite usually includes the tests related to the specific system of a release. QMTest saves the test results for further inspection with the graphical or command-line interface.

NICOS relies on the configuration management tool for driving QA and unit tests in the packages of a release. In ATLAS software release these tests are performed by `make` command with special targets. CMT tool is responsible for broadcasting these make commands to the packages. Integration tests are supposed to be represented by scripts with certain suffixes that could be run in the standard environment of the release. NICOS offers three options for description of the test scripts locations:

- Tests scripts are located in the single directory that is indicated in the NICOS configuration file.
- Test scripts are selected from the specifically named directories of packages (e.g. `test`). The directory name is indicated in the NICOS configuration file.
- The names of tests and locations are indicated in NICOS tests database file.

NICOS verifies tests results by checking the exit value returned by the test executable (should be zero for success) and by searching for certain text patterns in the test output. There are two levels of alarm: *warning* and *error*. The patterns of three types can be specified in nicos configuration file for each type of tests:

- *Success* pattern is required in the output of successful test. The absence of *success* pattern is an error sign.
- *Warning* pattern indicates “small” problems.
- *Error* pattern indicates “larger” problems.

Additional methods of test evaluation, such as regression testing and histogram comparison, can be added with external tools.

The packages with warnings or errors are highlighted with yellow and red colors on the NICOS web pages. The e-mail notification can be send to the authors of these packages if desired (both warnings and errors or only errors can trigger messages). The results on the web page with the integration test summary are similarly marked. The contact persons for the integration tests are listed in the NICOS test database file. The developers can personalize the test configuration by including the configuration parameters in the specially marked lines of the integration test scripts:

- Additional *success*, *warning*, *error* patterns.
- Additional e-mail addresses for notification.
- Name of test suite.

Sometimes minor modifications or bug fixing are performed for the software release that do not require a rebuild. The modular organization of NICOS allows to repeat tests and refresh the documentation skipping all other steps of the nightly builds.

STATUS AND PLANS

NICOS nightly builds are now widely used for the preparation of ATLAS stable software releases. The quality of the releases is verified by QA tests and 35 integration tests. The long term plans center around improving web page appearance and better control of test jobs (e.g. control of the output size of a test).

CONCLUSIONS

The NICOS nightly control tool provides a framework for tests of different types: quality assurance (or static), unit, integration. NICOS contains the mechanisms for the evaluation of test results which can be extended by plugging in the external tools. The tests results are immediately delivered to developers via automatic e-mails and NICOS web pages.

ACKNOWLEDGMENTS

The author wishes to thank members of the ATLAS Software Infrastructure team, the LCG SPI group and Physics Applications group at BNL for many valuable advices and useful discussions

REFERENCES

[1] A.Undrus, CHEP’03, La Jolla, USA, 2003, eConf C0303241, TUJT006 [hep-ex/0305087]. The NICOS home page is http://www.usatlas.bnl.gov/computing/software/nicos/index.html

[2] http://atlas.web.cern.ch/Atlas

[3] http://www.cmtsite.org

[4] http://www.codesourcery.com/qmtest

[5] http://lcgapp.cern.ch/project

[6] S. Albrand, “The Tag Collector. A Tool for Atlas Code Release Management”, CHEP04 contribution

[7] A. Vaniachine et al, CHEP’03, La Jolla, USA, 2003, eConf C0303241, MOKT006 (2003) [cs.db/0306103]; paper 212, these proceedings.

[8] http://sourceforge.net/projects/cppunit

[9] ATLAS software archirecture web page http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture