ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA049

ARES: AUTOMATIC RELEASE SERVICE

I. Prieto Barreiro, F. Varela. CERN, Geneva, Switzerland.

Abstract

This paper presents the *Automatic RElease Service* (ARES) developed by the Industrial Controls and Safety systems group at CERN. ARES provides tools and techniques to fully automate the software release procedure. The service replaces release mechanisms, which in some cases were cumbersome and error prone, by an automated procedure where the software release and publication is completed with a few mouse clicks. ARES allows optimizing the time and the work to be performed by developers in order to carry out a new release. Consequently, this enables more frequent releases and therefore a quicker reaction to user requests. The service uses standard technologies (Jenkins, Nexus, Maven, Drupal, MongoDB) to checkout, build, package and deploy software components to different repositories (Nexus, EDMS), as well as the final publication to Drupal web sites.

INTRODUCTION

At CERN, the Industrial Controls and Safety group of the Beams Department (BE-ICS) provides solutions and support for industrial control systems and develops, installs and maintains safety systems. The software implemented by the group covers all layers of the control systems, i.e. ranging from front-end devices like PLCs (*Programmable Logic Controllers*), Front-End Computers executing real-time tasks up to the SCADA HMI and web visualization. Multiple languages and packages are employed to develop all this software, like:

- The Siemens/ETM WinCC Open Architecture (OA) SCADA(*Supervisory Control And Data Acquisition*) product for the supervisory software.
- Unity Pro and Simatic Step7 for Schneider and Siemens PLC programming respectively.
- C++ for real-time applications, industrial middleware like OPC Unified Architecture and extensions to the commercial WinCC OA package.
- Java and Python for controls applications automatic generation tools.
- Java, Javascript and Angular JS for web visualizations.

In addition, due to the different operating systems used for operation in different domains at CERN, most of the software needs to be built for both MS Windows and Linux.

Although all software is committed to a common repository, the build and release processes were traditionally left up to each developer. This led to a very heterogeneous set of release mechanisms that included standard automated tools like Apache Maven [1], custom scripts (Python or Perl) which were maintained by the developers, or manual procedures consisting of multiple steps. In some cases, the release procedure was cumbersome and error prone leading to various problems:

- Very specific procedures required an expert knowledge in order to make a release of a software component. This represented a major problem when an urgent hot-fix was required in operation but the release expert was not around.
- Software releases were infrequent since developers would tend to include multiple bug fixes in a new version of the component to avoid releasing multiple times. This resulted in a long time for users to wait for a bug-fix.
- Manual steps led to incorrect information during the distribution of the components, e.g. inconsistency between version number in the web pages used for distribution and the component itself, which created confusion among the users as well as it caused some published versions to be unnoticed since the version number in the distribution pages had not been updated.

To avoid these issues, the BE-ICS group decided to develop ARES, which is described in the next sections.

AUTOMATIC RELEASE SERVICE

ARES makes use of standard technologies (Jenkins [2], Nexus [3], Maven, Drupal [4], MongoDB [5]) to provide a unified and automated release procedure, thus reducing the complexity and time required for releasing new software and keeping in sync the repositories with the software published for the final users. The following sections will describe the service architecture and the software release and publication workflows.

ARCHITECTURE

Figure 1 shows the main architecture and workflow of the release service. The software responsible persons use Jenkins as the web interface to trigger and complete the release. The build and release steps run in the background and are implemented using Apache Maven. This approach hides the complexity of the different steps involved in the release and allows to trigger new releases by non-expert users.

ARES uses two different repositories:

- Sonatype Nexus is the main software repository. It contains the software binaries plus additional metadata used to qualify the release, like the description of the target system for the software package.
- EDMS (*Electronic Document Management System*) [6]. The repository is used to store the software documentation and additional information like the software release notes.

Once the software has been released, the publication procedure uses Nexus, EDMS and Jira [7] as data sources to obtain the list of released versions, the documentation and

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA049

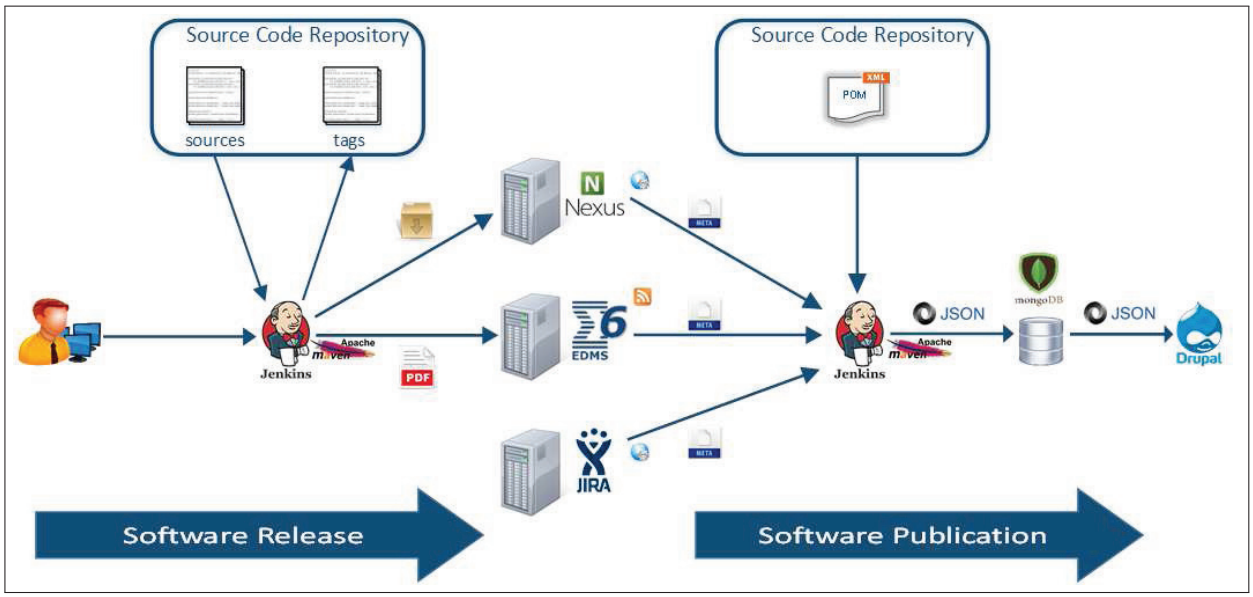

Figure 1: Service architecture.

the list of issues fixed in the new version of the component. The compiled information is stored in a non-relational database (MongoDB) where it is queried periodically by the web content management tool (Drupal).

Apache Maven plays an important role in the release and publication workflows, its main benefits will be explained in the next section.

Apache Maven

"Apache Maven is a software project management and comprehension tool. Based on the concept of a Project Object Model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information."

There are several benefits in using Maven in ARES:

- Definition of a common parent POM file. The software components having a similar nature, i.e. PLC baselines or SCADA packages, can share a parent POM file where all common tasks are defined. Examples of these tasks are: how to filter and package the source files, definition of the repository where to deploy the packaged software or the definition of the different release profiles.
- Release profile definitions. Maven allows to create different profiles to customize how the project is built and released. For example, ARES defines three different profiles for public releases, beta releases and snapshot releases. The main difference in the profiles is the repository where the artifacts are deployed, being internal repositories for the beta or snapshot releases. The public release profile deploys the software to staging repositories - explained in the following sections. The different profiles are activated automatically from Jenkins by parsing the version number of the release. For example, version numbers containing alphabetic characters (like '1.0-beta-01') will activate the beta release profile and version numbers containing only numbers (like '1.0') will activate the public release profile.
- Implement additional functionalities by creating new Maven plugins. In some cases, it might be necessary to incorporate additional features to a release build. Maven allows the implementation of plugins to provide additional goals that can be attached to a lifecycle phase. For example, ARES provides a plugin to interact with EDMS for creating new documents, new document versions and attaching files to the documents.

RELEASE WORKFLOW

The workflow used to release the software and documentation to the repositories is shown in Fig. 2.

Release Build

The release build is triggered from Jenkins by the responsible person of the software component. A release is a paremeterized build where the developer must introduce, at least, the release version for the software. In some cases it is also required to include additional meta-data like the target platform version. For example, the software packages for WinCC OA must specify the SCADA version the software is produced for.

Once the release build is triggered, ARES checks out the source code from the repository (SVN) and executes the Maven lifecycle phases [8] until the software is packaged and installed in the local repository. Some of the typical phases of the release build are the validation of the Maven project, compilation of the source code, execution of the unit tests, packaging of the compiled code into its distributable format, execution of integration tests and installation of the package into the Maven local repository.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA049

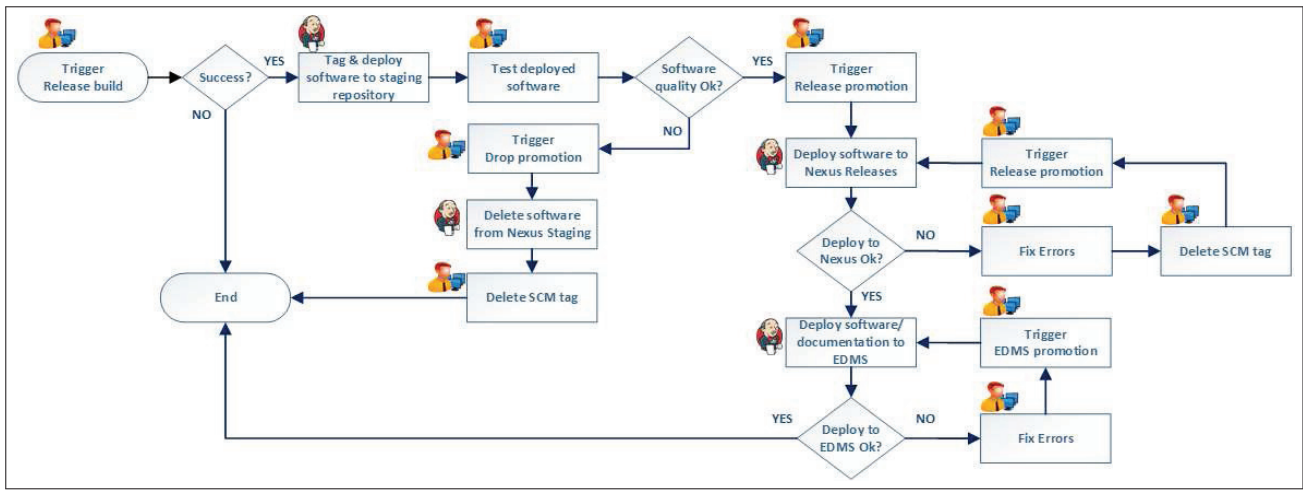

Figure 2: ARES release workflow.

Package Deployment

If the release build succeeds, the Maven Release Plugin [9] tags the source code in SVN and deploys the packaged software to a staging repository in Nexus. When the software is deployed to a staging repository it will not be available for the end users. Instead, the release will be kept in a private repository waiting for the validation of the integration tests or the quality assurance team.

Promotions

At this stage there are two possible outcomes for the software package:

- If the integration tests failed or the software did not meet the expected quality it can be dropped from the staging repository (drop promotion). In this case the tagged sources must be removed from the source code management system.
- Otherwise the staged software can be promoted and moved to a public repository (release promotion).

Both promotions are triggered from Jenkins and the only manual action to be performed by the software responsible is to delete the tagged sources in case of a drop promotion.

If the release promotion is executed there is an additional step to be performed: the deployment of the software documentation to EDMS. The deployment of the documentation is achieved using the EDMS Maven plugin [10] implemented by the BE-ICS group. The plugin provides goals to interact with EDMS and allows to create new documents and new document versions, add files to the document and change the document status (released, cancelled, obsolete, etc.).

In some cases, the deployment of the software or the documentation to the repositories might fail. For example, if the software artifact was promoted manually from Nexus or if the expected document version already exists in EDMS. In these situations, ARES uses the Build Failure Analyzer plugin [11] for Jenkins to analyze the error log and tries to map the error with a solution described in an internal knowledge database.

PUBLICATION WORKFLOW

At this stage the software package and documentation are deployed to a software repository and to a document management system. The end users can already download them directly from the repositories. However, it is not possible to customize the web pages of the repositories to provide a good user experience when looking for the relevant software. For this reason, each software framework provides its own website containing the different components, which includes the download links, the documentation, the release notes, additional information, etc. The JCOP [12] and UNICOS [13] frameworks use Drupal as a web content management and ARES ensures that the released software is always up to date and available for the end users. Figure 3 shows the software publication workflow.

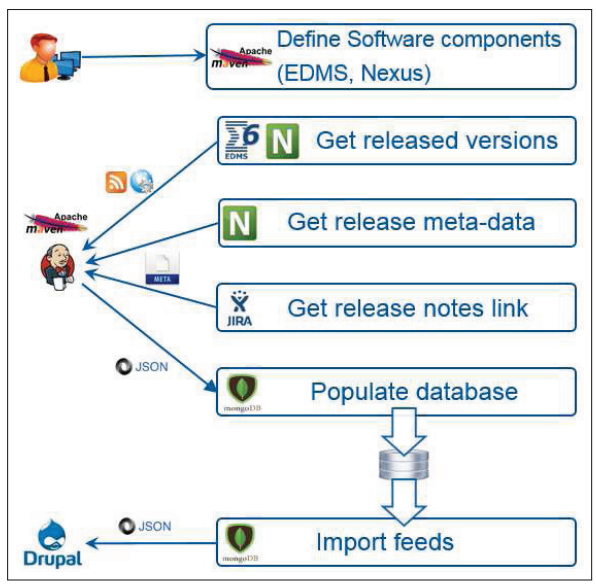

Figure 3: Software publication workflow.

The following sections explain the workflow details: the definition of the components relevant to the service, the

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA049

retrieval of the released versions from the sources and their publication in the framework webpages using Drupal.

Software Components Definition

Only a portion of the software deployed to the repositories is relevant for the publication in the frameworks websites. The service responsible must define what are the suitable components to look for new releases. In principle, it would be possible to store the information about the new versions in a database whenever the release is completed but this approach will only work for new releases, thus ignoring the software already available in the repositories. Instead, the approach implemented by the service is to define the list of relevant software components and to query a data source regularly to get the complete list of released versions.

This is achieved by using the Released Components Maven plugin [14] implemented by the BE-ICS group. The plug-in input is a POM file where the list of relevant components is defined together with some additional information:

- Data source for the releases. The possible data sources currently implemented by the plugin are Nexus and EDMS.
- The list of EDMS files that will be displayed next to the software download link (optional).
- The project identification in Jira (optional). This data will be used to obtain the Jira release notes.
- A list of released versions to ignore (optional). Useful when a released version became obsolete and it should not be used in any project.

Get the Released Versions

Once the list of components is defined it is possible to create Jenkins jobs to fetch the list of released versions regularly. The get-released-components goal of the Maven plugin will execute the following actions:

- Get the list of released versions from Nexus and EDMS for each software component. This is achieved using the Nexus REST API or the EDMS RSS feeds depending on the data sources defined for the component.
- Get the release meta-data from a properties file in Nexus.
- Get the release notes link from Jira.
- Store or update the release data in the MongoDB instance.

Drupal Publications

Finally, it is necessary to import the data from the database to Drupal making the releases visible for the end users. This step is achieved by exposing the data through a REST interface in MongoDB and using the Drupal feeds importer module. The imported data will then be mapped to a Drupal content type. ARES defines two different content types:

- Component Page. This content type defines the relevant data to identify a software component, like the component name, the person responsible for the software, the type of software (framework, component, subcomponent), etc.
- Component Release. A component release will always be linked to a unique component page. In this case, the content type defines the relevant data to a specific release, like the version number, the release date, the target platform, the software download link, the release notes link or the additional documentation files.

With the content types defined it is now possible to define the data mapping for the Drupal feeds importer. This configuration step consists on linking the fields obtained from the REST query to the database, in JSON format, to the fields available in the component release content type.

Before triggering the feeds importer it is also required to create the component pages for each software component. Once this step is completed, the feeds importer is able to import the data of the released software and link each release to a component page.

The visualization of the software releases in a component page is implemented using Drupal views. The views module allows defining the fields and the format for displaying the data available in the Drupal database (Fig. 4).

CONCLUSION

The BE-ICS group has made an effort in developing ARES and integrating the JCOP and UNICOS framework components in the new release service. Currently over a hundred components are integrated in the service, including WinCC OA components and PLC baselines. The number of software versions released by ARES is around a hundred and twenty for the current year and around three hundred and forty since the creation of the service in March 2016.

The software responsible persons profit from a simple release procedure executed from a unique web interface. The procedure allows to release more frequently and keeps in sync the list of published versions with the ones available in the repositories. As a consequence, the end users of the software components profit from the frequent releases and the coherence of the versions published in the framework websites.

In the coming months ARES will be adapted to use GIT as a version control system and the deletion of the source code tags in the release workflow will try to be automated.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA049

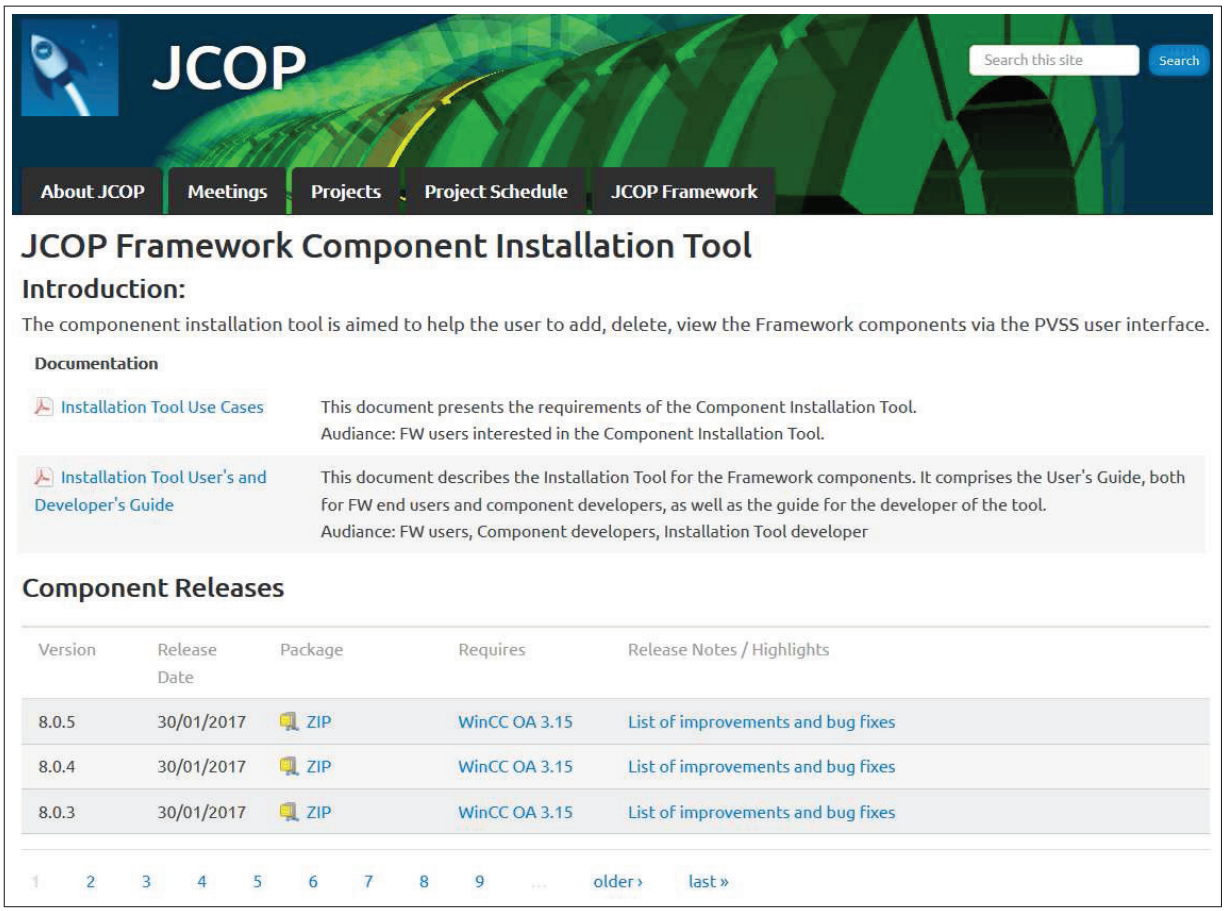

Figure 4: Visualization of a software component releases.

REFERENCES

[1] Apache Maven Project,
http://maven.apache.org/index.html

[2] Jenkins Continuous Integration and Continuous Delivery,
https://jenkins.io

[3] Sonatype Nexus
http://www.sonatype.org/nexus/

[4] Drupal
https://www.drupal.org

[5] MongoDB
https://www.mongodb.com

[6] Electronic Document Management System (EDMS)
https://edms.cern.ch/

[7] Atlassian Jira
https://www.atlassian.com/software/jira

[8] Introduction to Maven lifecycle
https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

[9] Maven release plugin
http://maven.apache.org/maven-release/maven-release-plugin/

[10] EDMS Maven plugin
https://edms-maven-plugin.web.cern.ch/

[11] Build Failure Analyzer plugin for Jenkins
https://wiki.jenkins.io/display/JENKINS/Build+Failure+Analyzer

[12] Joint COntrols Project (JCOP)
http://jcop.web.cern.ch

[13] UNified Industrial Control System (UNICOS)
http://unicos.web.cern.ch

[14] Released Components Maven plugin
https://released-components-maven-plugin.web.cern.ch