SYNCHRONISING LabVIEW DEVELOPMENT AND DEPLOYMENT ENVIRONMENT

O. O. Andreassen, C. Charrondiere, M. Miskowiec, H. Reymond, A. Rijllart
CERN, Geneva, Switzerland

ICALEPCS 2019

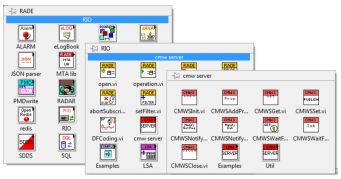
ABSTRACT

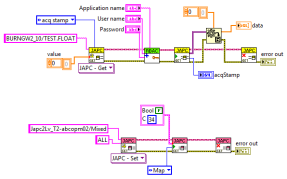
LabVIEW™ with its graphical approach is suited for engineers used to design and implement systems based on schematics and designs. Being a graphical language, it can be challenging to keep track of drivers, runtime engines, deployments and configurations since most of the tools on the market aimed towards this are implemented for textual languages. Configuration management is possible in the development environment via version control systems such as Perforce™, however at CERN and in the open source software development community in general, the tendency is moving towards Git.

In this paper we demonstrate how the combination of automated builds, packaging, versioning and consistent deployment can further ease and speed up development, while ensure robustness and coherency across systems. We also show how an in-house built tool called "RADE Installer" synchronises both development environments and drivers across workstations, empowering graphical development at CERN, by merging the open source toolchains with the workflow of LabVIEW™. RADE installer represents a solution for LabVIEW™ to keep track of drivers, runtime engines, deployments and configurations.

BACKGROUND

The **RADE** (Rapid Application Development Environment) palette is the solution we offer at CERN to develop expert tools, machine development analysis and test facilities, integrated with the CERN control infrastructure.

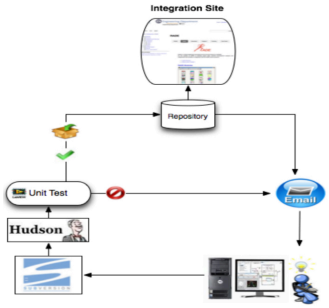


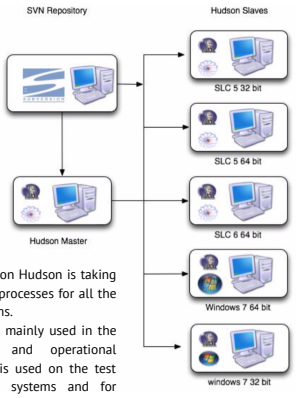


An example of simple GET data function used to readout values from a device, within the CERN control infrastructure.

An example of simple SET parameters function used to change the setup of a device within, the CERN control infrastructure.

With the increase of dependencies, complexity and the needs for quick prototype delivery, we started to invest in build automation and **Continuous Integration**. The release process was reduced from one day to about one hour.





The **Build Engine** based on Hudson is taking care of the compilation processes for all the RADE supported platforms.
Scientific Linux CERN is mainly used in the control infrastructure and operational environment. Windows is used on the test benches, local control systems and for analysis tools.

SYSTEM ARCHITECTURE

New CERN constraints
Market survey

Version Control

Continuous Integration server

Accelerators Control System OS

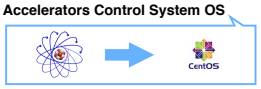


Table 1: Evaluation of package managers

Name	License	Local Server	Cross Platform	Mixed Language
VIPM [8]	Paid	Paid	Yes	No
NIPM [9]	NI	Yes	No	Yes
OPKG [7]	MIT	Yes	No	Yes
Yum [7]	GPLv2	Yes	No	Yes
Pip [7]	MIT	Yes	Yes	No
Dpkg [7]	GPLv2	Yes	No	Yes
Conda [7]	BSD	Yes	Yes	Yes

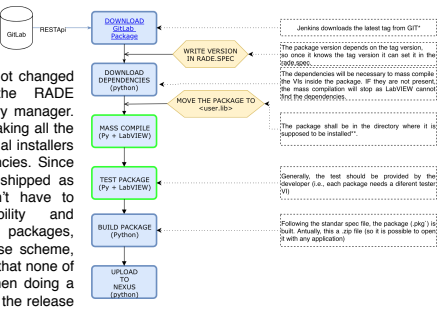
After evaluating several different commercial and open source products (see Table 1), it would initially seem like many of the package managers on the market could be suitable for our needs, however any cross-platform tool that was close to offering what we wanted, either would add a higher cost than the added value or it would be tailored to its dedicated environment and therefore not suitable for the job.

After a few internal review rounds, evaluating tools and looking at the results forming from the data, we concluded with that the best path for us to go, fully being able to control and adapt the environment to our need, would be to make a custom-tailored tool that could hook in to other existing technologies and the same time being compatible with existing environments, ensuring compatibility and reducing risk of cross pollution between installers.

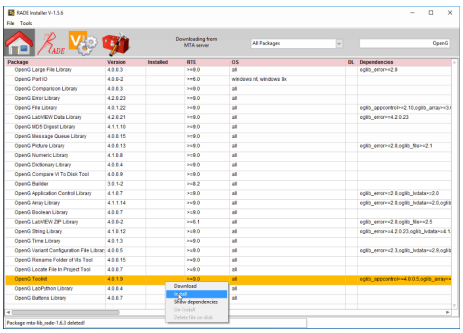
Table 2: Evaluation of repository tools

Name	License	API	Mixed Package	Server-Side Management
VIPM [8]	Paid	No	No	No
EOS [13]	N/A	No	Yes	No
GitLab [10]	MIT	Yes	Yes	Yes
WebSite	N/A	No	Yes	No
Nexus [14]	GPL	Yes	Yes	Yes
PyPI [15]	GPLv2	Yes	No	No

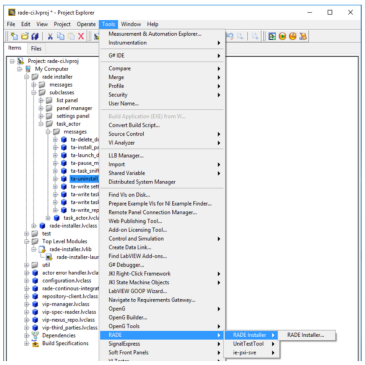
After evaluating and testing several repository tools, some of which are shown in Table 2 above, we narrowed the list down to 2 possible candidates that would suite our needs: Sonatype Nexus and GitLab. At the time the comparison was done, GitLab was not initially intended to be a repository solution, but as time has gone by, more and more features have been added and it is now (2019) a full "DevOps" solution that supports the whole software workflow, therefore, given the features available we went decided to use Nexus as our repository host.



The RADE build cycle has not changed much after introducing the RADE installer and Nexus repository manager. The main challenge was breaking all the different libraries into individual installers and map their interdependencies. Since everything in the past was shipped as one big package, we didn't have to manage the interoperability and compatibility between packages, however with the new release scheme, we always have to take care that none of the libraries break or fail when doing a release. As an added bonus, the release time has gone down even more, and we can now release stable packages within minutes and add new packages incrementally without affecting users.



The RADE Installer can be launched from within the LabVIEW™ development environment. An option available from the "Tools -> RADE" menu will launch a statically compiled version of the installer, built using packed project libraries. This allows for instant installation of a desired package without having to restart LabVIEW™. Only in cases where a system package requiring restart of the operating system does one have to restart the environment.



The RADE installer was implemented with ease of use and efficiency, in terms of workflow, in mind. The developer should not need to focus on versioning and inter module compatibility, rather get the necessary dependencies installed with as little interactions as possible. If you check out an existing project from git that contains any package in the RADE eco system, you only have to reference the project and the installer will find the implicit dependencies. Any update and new package are visible from the interface, and the user can toggle between "public", "RADE" and "VIPM" packages. The user can also choose if updates should be installed automatically (hidden) or manually.

CONCLUSIONS

Introducing Nexus and the possibility to break packages into singular elements in the RADE CI engine has greatly improved our capacity to both release and keep track of packages. Cross dependent developments using both Java, C++ and LabVIEW™ have benefitted from the new structure and we have become more conscious in designing packages with test and traceability in mind. The introduction of the RADE installer in the team is still an ongoing process, but the benefits outweighs the efforts so far.

It has become easier to share and reuse code, and adding packages to the installer encourages a workflow that reduces errors in deliveries.The RADE installer still does not support full environment installation, so the plan is to add this functionality in the next release. We also have to follow closely what National Instruments plans to do with their Next Generation environment and make sure the changes we do are compatible with future releases

EN ENGINEERING DEPARTMENT
LABVIEW.SUPPORT@CERN.CH