THE LABVIEW RADE FRAMEWORK DISTRIBUTED ARCHITECTURE

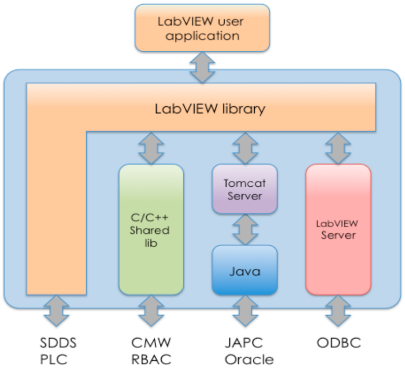
O. Ø. Andreassen, D. Kudryavtsev, A. Raimondo, A. Rijllart, V. Shaipov, R. Sorokoletov, CERN, Geneva, Switzerland

ABSTRACT

For accelerator GUI applications there is a need for a rapid development environment to create expert tools or to prototype operator applications. Typically a variety of tools are being used, such as Matlab or Excel, but their scope is limited, either because of their low flexibility or limited integration into the accelerator infrastructure. In addition, having several tools obliges users to deal with different programming techniques and data structures. We have addressed these limitations by using LabVIEW, extending it with interfaces to C++ and Java. In this way it fulfils requirements of ease of use, flexibility and connectivity, which makes up what we refer to as the RADE framework. Recent application requirements could only be met by implementing a distributed architecture with multiple servers running multiple services. This brought us the additional advantage to implement redundant services, to increase the availability and to make transparent updates. We will present two applications requiring high availability. We also report on issues encountered with such a distributed architecture and how we have addressed them. The latest extension of the framework is to industrial equipment, with program templates and drivers for PLCs (Siemens and Schneider) and PXI with LabVIEW-Real Time.

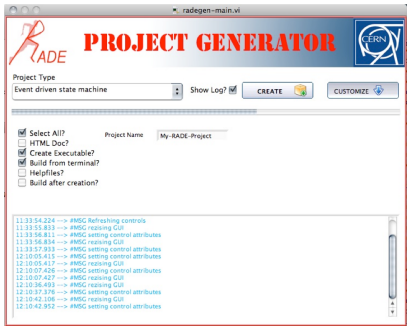
Distributed Architecture

RADE (Rapid Application Development Environment) is a collection of software libraries providing a defined application-programming interface (API). With the RADE framework and LabVIEW we have managed to ease the job of integrating new and existing projects into all the different disciplines at CERN. With our distributed architecture the user does not have to update their implementation, we maintain them on our server and ensure compatibility.



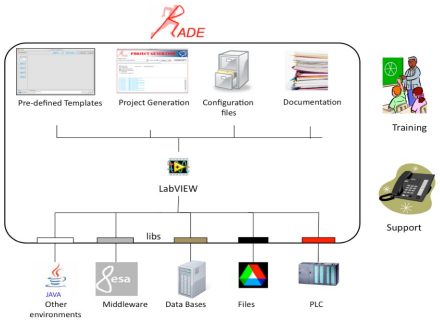
RADE LIBRARIES

The RADE libraries consist of several communication layers adapted to the necessary protocols. It also makes use of a distributed architecture where several redundant servers host the communication and analysis libraries. These can potentially be written in any software language, but we mainly use the CERN Java packages and native LabVIEW servers that communicate with the LabVIEW client through sockets.



Total Package

The RADE framework aims to give users a total package for development, maintenance and support, making it quick to implement yet highly stable, maintainable and flexible through well defined development templates, guidelines and documentation.

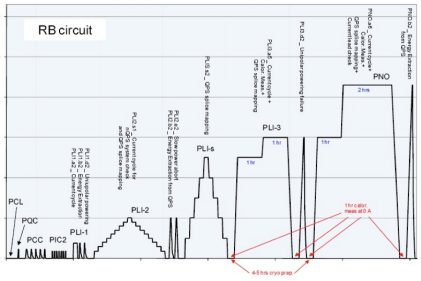


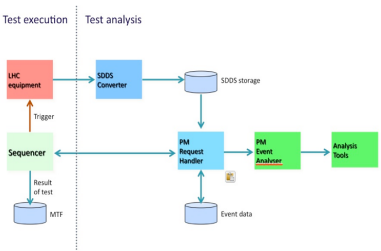
RADE in Hardware Commissioning

Hardware Commissioning is the phase where tests are carried-out by the specialized teams to qualify the individual systems of the LHC for operation. The post-mortem system has the role of grouping and analysing transient data recorded in the LHC equipment. With RADE the Post Mortem Framework is highly flexible. In 2011, 6092 tests where executed and approved by software using RADE.

RADE in the LHC Dashboard

The LHC Dashboard project aims to provide a single entry point to the monitoring data collected from the LHC and it equipment. The Dashboard collects information from multiple sources, treats the data and renders it as a set of web pages. It covers various activities of the LHC making up a complete picture of what goes on in the machine. By using RADE in the dashboard we have managed to make a very flexible and powerful online- data presentation tool.





CONCLUSION

The flexibility and short development time experienced when using RADE in, amongst others, the presented project examples shows that the RADE framework is an excellent and powerful tool that can be used to cope with challenges in an environment that quickly and constantly changes. One has however to take care when using a distributed architecture since introduction of new “features” or changes can quickly cause unforeseen problems that affect many users. On the other side the flexibility and convenience of having a distributed architecture trumps the downsides by far.