FESA3 INTEGRATION IN GSI FOR FAIR

S. Matthies, H. Bräuning, A. Schwinn, GSI, Darmstadt, Germany
S. Deghaye, CERN, Geneva, Switzerland

Abstract

GSI decided to use FESA (Front-End Software Architecture) as the front-end software tool kit for the FAIR accelerator complex. FESA was originally developed at CERN. Since 2010 FESA3, a revised version of FESA, is developed in the frame of an international collaboration between CERN and GSI. During development of FESA3 emphasis was placed on the possibility of flexible customization for different environments and to provide site-specific extensions to allow adaptation for the contributors. GSI is the first institute different than CERN to integrate FESA3 into its control system environment. Some of the necessary preparations have already been performed to establish FESA3 at GSI. Examples are RPM packaging for multiple installations, support for site-specific properties and data types, first integration of the White Rabbit based timing system, etc.. Further developments such as e.g. integration of a site-specific database or the full integration of GSI's beam process concept for FAIR will follow.

INTRODUCTION

GSI's FAIR [1] project is a challenge and a chance to establish a revised control system solution. A couple of years ago it was decided to develop the main parts of the future control system for FAIR (such as FESA, LSA [2] and the middleware CMW [3]) in the frame of an international collaboration with CERN.

This paper gives an overview of how the FESA3 framework is extended to suit into the future FAIR control system environment.

MODULARITY OF FESA3

To establish the FESA framework on sites different than CERN the main focus during development of FESA3 was modularity and extensibility. Modularity is achieved by clear separation of its components and involved technologies into core- and site-specific packages. Extensibility of the FESA3 framework is ensured by the possibility to provide site-specific extensions to the core packages. This involves the FESA3 framework packages as well as the components of the FESA3 plug-in for the integrated development environment Eclipse.

In general the core packages contain the common code base that is used by both participating sites. The common part provides the interfaces, (abstract) base classes as well as the functionality that does not have to be extended.

The FESA3 framework combines the usage of different technologies and programming languages such as XML, XSLT, Python, C++ and JAVA.

The site-specific components extend the common part by providing the functionality required only by the implementing site. This is realized by using software design concepts such as inversion of control and inheritance, depending on the technology used. Figure 1 gives an overview of the main FESA3 framework components. The extension packages are marked by "-EXT" which stands for either CERN or GSI. Accordingly a similar structure is realized for the parts that constitute the JAVA based FESA3 Eclipse plug-in.

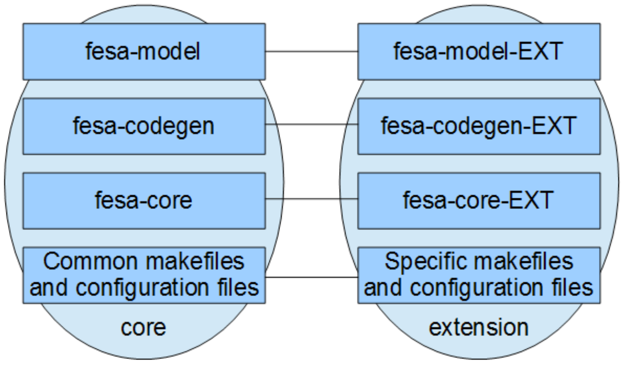

Figure 1: FESA3 Framework Components.

SITE-SPECIFIC FESA FEATURES

The FESA3 installation at GSI has several site-specific features and extensions.

Standard Properties

To provide a common interface of FESA3 based equipment software to the application layer, an elaborated set of standard properties is defined at GSI. Standard properties are common properties that each accelerator device should provide to the application layer. Typical examples are properties such as Status, Power, Init or Version. Site-specific properties may be coupled to site-specific data field types. The properties as well as their data fields are pre-defined in the site-specific template for new FESA3 equipment software. For operational FESA based equipment software to be used within the FAIR control system environment these properties must be implemented by the FESA equipment software developer. GSI's FESA development guideline outlines this and the other issues to be considered when developing productive FESA3 based equipment software for operation of the FAIR accelerator complex.

ISBN 978-3-95450-146-5

FAIR Timing Integration

For FAIR a newly build timing system will be used [4].

To be able to react to timing events sent via the new timing network FESA3 provides a prototype integration of the FAIR timing system. This is achieved by providing an additional timing event source within the site-specific part of the FESA core framework. The timing event source analyses the incoming FAIR timing events and triggers the execution of FESA real-time actions. FESA real-time actions are basically threads that can be run in "real-time" and do not directly interact with a client.

Multiplexing: GSI's Beam Process Concept

For operating the FAIR accelerator complex, when several experiments are supported with different beams on a pulse to pulse basis, a staged multiplexing concept will be used. The basic element for describing the actions in the machines is the Beam Process. A Beam Process is an uninterruptible activity like a beam injection, acceleration or beam transfer. Beam Processes are combined into Sequences. A Sequence may represent, for example, a synchrotron accelerator cycle, a set of beam manipulations in a storage ring or the transfer of a beam from one machine to another.

Depending on its functionality, a device may either have different settings for each Beam Process or have one setting for each Sequence, which means it behaves identically for each Beam Process in the Sequence.

This concept has an impact on the existing FESA3 implementation of multiplexing that is tailored to what was needed so far. Since there was no huge difference between CERN's and GSI's requirements the implementation was located in the core part of the FESA framework. To realize GSI's concept of multiplexing using Beam Processes or Sequences in accordance to CERN's multiplexing concept the implementation of the cycle selectors for both involved sites must be moved to the site-specific part.

FESA3 INSTALLATION

One of the specific requirements of GSI is to support installations of FESA3 on multiple Scientific Linux machines. Examples are the servers of GSI's local development cluster, the control system installation for the Proton Linac Source that will be constructed at CEA[5] in Saclay, France or the development environment for the Slovenian in-kind contribution for FAIR. This is in contrast to CERN's need to provide a FESA3 installation within a single development environment for local FESA developers only.

To easily support multiple FESA3 installations on various Linux machines it was decided at GSI to provide FESA3 as well as the required CMW middleware and other site-specific 3rd-party libraries as an RPM (**R**PM **P**ackage **M**anager) based installation. This allows distributing FESA3 in a comparably simple way. The main advantage is that RPM packaging enables consistent and repeatable FESA3 installations on different Linux machines in varying environments.

This involves the realization of an installation directory structure that allows more than one FESA3 version in parallel. The chosen installation directory structure for FESA3 is to the greatest possible extent in accordance to a standard Linux directory structure.

EXTENSION OF THE USER INTERFACE: FESA3 ECLIPSE PLUG-IN

Since 2010 the graphical user interface for FESA3 is directly integrated in the development environment Eclipse as a plug-in. This allows to outline the basic development workflow of FESA3 based equipment software. The FESA3 development workflow for FESA equipment software developers involves steps such as

- creating / editing the FESA class design which is formally an XML document to outline the interfaces to the client and the equipment
- implementing automatically generated C++ source code frames which are based on the class design
- compiling the source code and linking the libraries
- creating device instances which are formally described in XML documents
- synchronizing the source code with a software repository such as SVN
- deploying the resulting software, configuration files and start scripts locally and remote on front-end computers
- testing the results locally and remote.

Figure 2 gives a basic overview of the typical development workflow for FESA equipment software.

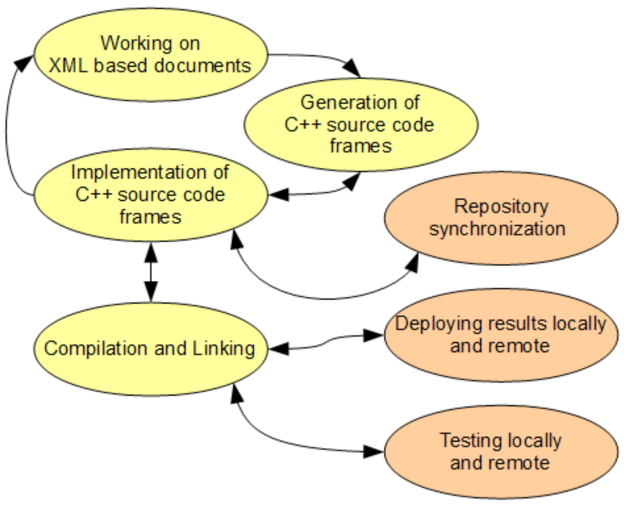
Working on XML based documents
Generation of C++ source code frames
Implementation of C++ source code frames
Repository synchronization
Compilation and Linking
Deploying results locally and remote
Testing locally and remote

Figure 2: FESA Development Workflow.

ISBN 978-3-95450-146-5

The integrated development environment Eclipse provides a solid base for such differing steps including the possibility for custom enhancements and extensions.

So far site-specific characteristics that affect the graphical user interface such as

- different FESA framework installation locations
- support of 32- and 64-bit CPU architectures
- an extensive and configurable delivery process and structure
- an extended and custom-tailored FESA test tool
- a differing SVN repository layout
- a different set of Eclipse cheat sheets to illustrate the recommended FESA software development workflow and provide direct access to further information
- varying automatically generated Make targets
- site-specific expert settings to help ease and speed up the development workflow itself
- a different FESA database environment

have been identified and integrated.

The design involved providing the possibility to extend the core parts by specific parts using software design concepts such as interface-based programming and inheritance. Wherever possible the varying site-specific parts are kept configurable. In addition, the missing flexibility concerning the SVN repository layout or the site-specific database integration is ensured by providing a set of specific JAVA packages.

TESTING IT ALL: THE CRYRING

The Swedish contribution to the FAIR project "CRYRING" serves as an early test system for the first assembly of FAIR's control system. The assembly is currently planned for the end of 2014. The intention is to couple the control system components LSA, CMW and FESA3 for operation in combination with a new FAIR timing system at GSI.

The commissioning of the CRYRING is the first time that the future control system components interact. The intention is to find out how the different parts add up with all the other components.

For this first assembly FESA3 based equipment software is developed at GSI by software developers and equipment specialists in different groups. As more and more developers use the FESA3 framework to produce operational equipment software the issues that require improvement can be identified.

COLLABORATION

The collaborative approach of developing the FESA3 framework is beneficial for all participating sites. Collaboration has its advantages for all intents and purposes when it comes to the integration of new concepts and ideas, improvements or the implementation of bug fixes.

However software development in collaboration increases the complexity of the whole development workflow itself. Particular care must be taken when introducing changes that certainly will affect the requirements of the other site. Mutual coordination on a regular basis is essential to keep both participating sites informed and up to date.

CONCLUSION

Since starting the collaborative development of FESA3 in 2010 several challenges have been mastered to establish FESA3 on a site different than CERN. These efforts not only include the technical issues mentioned in this paper but also constituting a development environment that involves comprehensive information and documentation for FESA3 equipment software developers.

In the past few years it has been shown that the flexible and modular approach of FESA3 supports the adaptation to different sites and environments.

The essential parts of the FESA3 framework have been established up to now. For the future several issues remain to improve the stability and usability of the FESA3 software at the involved sites.

REFERENCES

[1] FAIR website: http://www.fair-center.de
[2] J. Fitzek et al., "Settings Management within the FAIR Control System based on the CERN LSA Framework", WEPL008, PCaPAC'10, http://www.jacow.org
[3] V. Rapp et al., "Controls middleware for FAIR", WCO102, PCaPAC'14, http://www.jacow.org
[4] M. Kreider et al., "Launching the FAIR Timing System with CRYRING", TCO304, PCaPAC'14, http://www.jacow.org
[5] CEA website: http://www.cea.fr/english-portal