Table of Contents

CHEP 2012 Abstracts from SA2 at CERN

Multi-platform Automated Software Building and Packaging

A.Abad Rodriguez, L.Dini, A.Di Meglio, D.Meneses, V.Gouveia, A.Aimar

One of the major goals of the EMI (European Middleware Initiative) project is the integration of several components of the pre-existing middleware (ARC, gLite, UNICORE and dCache) into a single consistent set of packages with uniform distributions and repositories. Those individual middleware projects have been developed in the last decade by tens of development teams and before EMI were all built and tested using different tools and dedicated services. The software, millions of lines of code, is written in several programming languages and supports multiple platforms. Therefore a viable solution ought to be able to build and test applications on multiple programming languages using common dependencies on all selected platforms. It should, in addition, package the resultant software in formats compatible with the popular Linux distributions, such as Fedora and Debian, and store them in repositories from which all EMI software can be accessed and installed in a uniform way.

Despite this highly heterogeneous initial situation, a single common solution, with the aim of quickly automating the integration of the middleware products, had to be selected and implemented in a few months after the beginning of the EMI project. Because of the previous knowledge and the short time available in order to provide this common solution, the ETICS service, where the gLite middleware was already built for years, was selected.

This contribution describes how the team in charge of providing a common EMI build and packaging infrastructure to the whole project has developed a homogeneous solution for releasing and packaging the EMI components from the initial set of tools used by the earlier middleware projects. An important element of the presentation is about the developers experience and feedback on converging on ETICS and on the ongoing work in order to integrate more widely used and supported build and packaging solutions of the Linux platforms.

Elastic Testbed at CERN for the Integration of the EMI Middleware

T.Wolak, A. Elwell

The development and distribution of Grid middleware software projects, as large, complex, distributed systems require a sizeable computing infrastructure for each stage of the software process: for instance pools of machines for building, and testing on several platforms. Software testing and the possibility of implementing realistic scenarios for the verification of grid middleware are a crucial part of the testing process. System integration testing of a large number of components requires a large dedicated testing infrastructure installed and ready to host such tests. In the grid community such testing environment is described as a grid integration testbed . It is a dedicated grid infrastructure having similar organization, in smaller scale, as production installations where inter-component tests can be executed on different versions and platforms.

This contribution presents the implementation, based on elastic virtualized resources, of the grid testbed provided by the Grid Technologies group at CERN in order to support the developers of the DPM, FTS, LFC teams and the part of the EMI integration testbed hosted at CERN. The implementation of the EMI testbed also provides the integration of the Nagios monitoring probes of the installed services and supports several platforms such as Scientific Linux and Debian for 32 and 64 bits architectures. We will also present the lessons learned and the experience gained during the migration from the Linux Xen virtualization platform, used for gLite, to Microsoft Hyper V currently used for the EMI testbed.

Experiences with Software Quality Metrics in the EMI Middleware

M.Alandes Pradillo, E.Kenny, G.Pucciani, D.Meneses

The EMI Quality Model has been created to define, and later review, the EMI (European Middleware Initiative) software product and process quality. A quality model is based on a set of software quality metrics and helps to set clear and measurable quality goals for software products and processes. The EMI Quality Model follows the ISO/IEC 9126 Software Engineering Product Quality to identify a set of characteristics that need to be present in the EMI software. For each software characteristic, such as portability, maintainability, compliance, etc, a set of associated metrics and KPIs (Key Performance Indicators) are identified.

This article presents how the EMI Quality Model and the EMI Metrics have been defined in the context of the software quality assurance activities carried out in EMI. It also describes the measurement plan and presents some of the metrics reports that have been produced for the EMI releases and updates. It also covers which tools and techniques can be used by any software project to extract code metrics on the status of the software products and process metrics related to the quality of the development and support process such as reaction time to critical bugs, requirements tracking and delays in product releases.

Why Are Common Quality and Development Policies Needed?

M.Alandes Pradillo, L.Dini

The EMI project is based on the collaboration of four major middleware projects in Europe, all already developing middleware products and having their pre-existing strategies for developing, releasing and controlling their software artefacts. In total, the EMI project is made up of about thirty development individual teams, called Product Teams in EMI. A Product Team is responsible for the entire lifecycle of specific products or small groups of tightly coupled products, including the development of test-suites to be peer reviewed within the overall certification process.

The Quality Assurance in EMI (European Middleware Initiative), as requested by the grid infrastructures and the EU funding agency, must support the teams in providing uniform releases and interoperable middleware distributions, with a common degree of verification and validation of the software and with metrics and objective criteria to compare product quality and evolution over time. In order to achieve these goals the QA team in EMI has defined and now it monitors the development work and release with a set of comprehensive policies covering all aspects of a software project such as packaging, configuration, documentation, certification, release management and testing.

This contribution will present with practical and useful examples the achievements, problems encountered and lessons learned in the definition, implementation and review of Quality Assurance and Development policies. It also describes how these policies have been implemented in the EMI project including the benefits and difficulties encountered by the developers in the project. The main value of this contribution is that all the policies explained are not depending on EMI or grid environments and can be used by any software project.

This topic: EMI > CHEP2012SA2Papers
Topic revision: r2 - 2011-10-28 - AlbertoAimar