ISTITUTO NAZIONALE DI FISICA NUCLEARE

CNAF

INFN/CCR-11/02
September 1, 2011

A DATA ENVIRONMENT FOR SOFTWARE DEVELOPMENT PROCESS

Marco Canaparo , Claudio Galli, Elisabetta Ronchieri, Cristina Vistoli
INFN-CNAF, Viale Berti Pichat 6/2, I-40126 Bologna, Italy

Abstract

The maturity of software development process is strictly related to the implementation of the best practices typically followed by software team to perform particular tasks and to meet particular objectives. Its improvement is guaranteed by the presence of metrics that are designed and measured to plan and control productivity, effectiveness, quality and timeliness of software projects and products. The measurement of metrics contributes to collecting right data to the handling of the analysis process, and to establishing a dashboard to the management of the overall health of the process.

This paper describes a data environment suitable for improving the quality of the software process, developed in the context of the ETICS 2 European project. The data environment encompasses: 1) the trend analysis disseminator; and 2) the representation of software metrics and other useful software project information according to a standard organizational dashboard. The paper also describes the data environment implementations.

PACS: 89.20-Ff

Published by ***SIDS–Pubblicazioni***
Laboratori Nazionali di Frascati

1 INTRODUCTION

Customers expect to get software products that work without defects, that meet and even exceed their needs and expectations, and all these within continuously time-varying frames. In order to reach these goals, software development managers face the major challenge of improving products and processes by implementing tactics and methods that have been shown through real-life implementation to be successful. For example, the selection of project team members is a practice that will help define the ability of the team to be capable of rapid response: a cross-functional development team would be an example of a best practice [19]. Various research studies and design gurus have identified a large number of such best practices that can be beneficially implemented in the team's software development processes [6]. The performance of software processes and products is understood, tracked, controlled and predicted by metrics whose measurement allows software managers to handle continuously changing business conditions [15]. If metrics are to provide information, everyone involved in selecting, designing, implementing, collecting, and utilizing them must understand their definition and purpose [28].

Software quality is directly related to the quality of the process through which software is developed. Metrics are essentials in the assessment of the quality of software development processes by providing information about the development process itself over time. Measurement enables the software team to improve the software process, assist in planning tracking and controlling the software project. The analysis and monitoring of process performance are fundamental for improving the reaction time of a software team and effectiveness of its products. Several proposals can be found in the literature for collecting and storing process performance related data in order to support its analysis [20, 26]. In the context of software development process, data environment is an integrated data collection aimed at supporting decision-making processes [23]. It addresses the issue of arranging, integrating, representing and transforming project quantitative data to a unified and centralized organizational view. It also covers the issue of providing analytical functionalities for monitoring purposes.

In this paper we describe a data environment developed as part of the metrics program of the ETICS 2 European project ended in February 2010. ETICS 2 has provided a service to help software developers, managers and users to better manage complexity and improve the quality of their software. The ETICS 2 consortium has consisted of CERN (coordinator), INFN, Engineering Ingegneria Informatica S.p.A., 4D Soft Ltd., MTA SZTAKI, Vega IT GmbH, Forschungzentrum Julich, and the University of Wisconsin-Madison. Since May 2010 the ETICS 2 service has been included in the European Middleware Initiative (EMI) project (`http://www.eu-emi.eu/`). ETICS is an integrated infrastructure for the automated build, configuration, integration and testing of software developed by different teams widespread geographically. Up to now it has been used by Grid and non-Grid software[1] such as EGEE (`http://public.eu-egee.org/`), DILIGENT (`http://diligent.ercim.eu/`) and EMI. This infrastructure provides a service for software projects by integrating well-established existing procedures, tools and resources in a coherent infrastructure, and adapting them to the special needs of software projects. For example, a software project may require a dedicated resource to be compiled against a non-free thirdparty package. ETICS provides an intuitive access point through a Web portal and a professionally-managed multi-platform capability based on proven Grid technologies [1, 10].

Data environment has been designed and developed by CERN, INFN and Engineering Ingegneria Informatica S.p.A.. The core of data environment includes the trend-analysis disseminator and the dashboard monitoring system taking part of the ETICS architecture. Its goal is to provide a centralized and unified view of all software projects together with functionalities that easily allow software analysis on the base of available metrics according to different summarization levels. To achieve this goal two tools have been designed and developed: the ETICS Disseminator for trend-analysis that enables users to control the values of some metrics over time in simple and clear charts; the ETICS Dashboard for monitoring the compiling and testing results that enables users to have an instant view of trends, and anomalies in the software development activity. This paper describes

[1]Grid provides a set of services that allow a widely distributed collection of resources to be tied together into a computational framework.

the problems arisen and the solution found with the technical features of data environment and its current implementation.

The remainder of this paper is structured as follows: Section 2 provides an overview of the ETICS architecture; Section 3 details the ETICS Disseminator for trend analysis; Section 4 describes the ETICS Dashboard; Section 5 discusses related work; Section 6 point out future work; and Section 7 draws conclusions.

2 Overview of ETICS

ETICS is to provide a service for software projects by integrating well-established procedures, tools and machines in a coherent framework and adapting them to the special needs of software projects. It maintains and manages integrated pools of machines for running automated builds and test suites. ETICS has designed a data model on top of which its architecture has been built.

2.1 Data Model

The ETICS data model is designed to organize a generic software project by using the high-level entities such as the project internal structure, the software relationships and the operations required to build and test such a project [3]. The model is inspired by the Common Information Model (CIM) Application model[2] and the Object Management Group (OMG)'s Model Driven Architecture[3]. Nevertheless, it adds definitions for the operations of software construction (build) and verification (testing) which are missing from the mentioned models above. The ETICS data model is composed of several objects, which can be organized in software structures, and build and testing configurations. It explicitly describes the objects and the relationships between objects.

The software structure is characterized by the concepts of component, subsystem and project [2]: *component* is a portion of code providing a well-defined

[2]Common Information Model (CIM) Specification, 2.2, June 14, 1999, `http://www.dmtf.org/spec/cims.html`

[3]The Architecture of Choice for a Changing World, MDA, `http://www.omg.org/mda/`.

functionality within the system architecture; *subsystem* is a logical container of the overall architecture in more specific subsets of functionalities; and *project* is a container of well-defined high-level functionalities according to predefined user requirements. Our explanation of component is simplified compared to what is defined in reference [27]. A proper understanding of component requires an investigation of the concepts of code and functionality. The former can be at least a source file, or a configuration file or a document, whilst the latter is the sum or any aspect of what a software application can do for a user. Having said that, a subsystem is simply a logical container of components, whilst a project is a logical container of components and subsystems. For instance, a project can be composed of one or more components, one or more subsystems, or a combination of components and subsystems. Figure 1 shows the relations amongst project, subsystem and component entities.

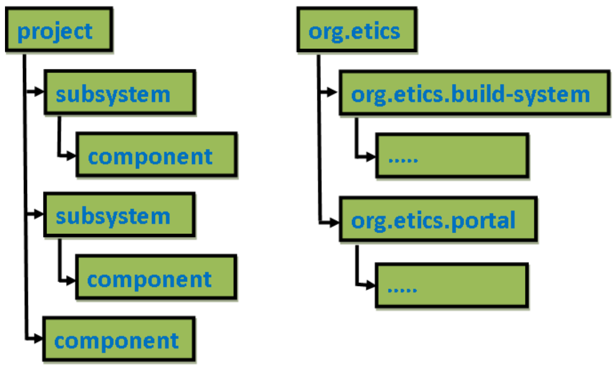


Figure 1: ETICS software project structure. On the left it is represented a generic software project structure, whilst on the right a partial software structure of the `org.etics` project.

Moreover, the general term *module* is used for referring to any of project, subsystem and component. Each module holds at least one configuration.

The build and testing configuration is composed of platforms, commands, properties, environment variables and dependencies: *platform* contains operating system, architecture and compiler information, as for example `slc4_x86_64_gcc345`

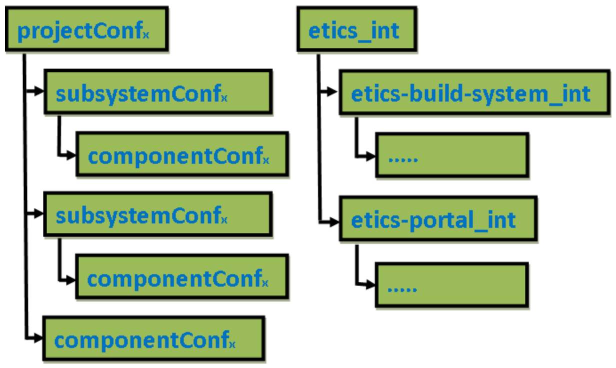


Figure 2: ETICS project configuration structure. On the left it is represented a generic configuration project structure, whilst on the right a partial configuration structure of the project `etics_int` configuration belonging to the `org.etics` project. Moreover, on the left the x index shown in the configuration name explains that each module (either project or subsystem or component) holds at least one configuration.

where `slc4` is the operating system, `x86_64` is the machine architecture and `gcc345` is the compiler; *version control system commands* are used to checking out software on a local area; *build commands* are used to configuring, build, packaging, creating documentation, and removing generated build files (e.g., make clean, make doc); *test commands* are involved in running, for example, specialized unit test, coverage test, coding conventions test, functional test, stress test and performance test; *properties* are a set of custom attributes that a configuration requires at build-time such as compiling options; *environment variables* are a set of dynamic values that can affect the way running processes will behave; *dependencies* are optional link between component's configurations representing a software constraint. For each supported platform, a configuration contains information for checking out, compiling, and testing a subset of software as well as for handling software dependencies. Configurations can be linked one another in order to produce a tree whose root is a project's configuration and leaves are components' configurations.

Figure 2 shows an example of a given configuration project. The names in the green boxes represent the name of each configuration. Project and subsystem configurations are the only ones able to define different trees and subtrees respectively.

2.2 Architecture

The ETICS architecture is composed of several services as shown in Figure 3: Configuration Service, Repository Service and Execution Engine. The ETICS

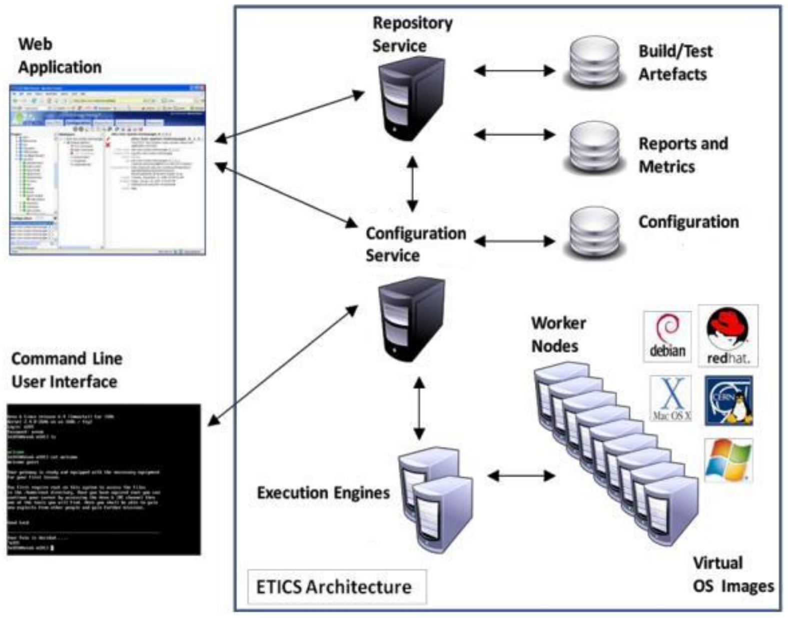


Figure 3: The ETICS architecture.

services specified in the following list are accessible to the ETICS communities via Command Line Interface (CLI) and Web Portal (WP). The ETICS CLI provides a similar functionality as the ETICS WP and makes use of the same Web service interface for simplicity and symmetry. CLI can be used directly by the user on local machines (e.g., a developer machine). Furthermore, the same client is used in an almost identical context by the Execution Engine. This similarity is crucial to avoid context switching between local and remote builds or tests[4], which would reduce the usability and reliability of the system.

The ETICS data are listed in three databases: the first is related to build or test artifacts, the second one to reports and metrics, and the last one to configurations.

The underlying security infrastructure is based on standard x.509 certificates [16]. Users are modelled as fully qualified x.509 principal names as they appear in standard x.509 -compliant certificates. The Web service verifies the user certificate Distinguished Name (DN) in the database of existing users involved in a project, and it allows or denies the operation according to the roles assigned to the users. From that point onwards, the Web service uses a service certificate to interact with other internal services. The access control list on the persisted data will be enforced by the Web service. The identified roles are described in the following list: administrator is a kind of super user enabled to perform all the operations allowed in the ETICS infrastructure; module administrator is responsible for handling the project by using the ETICS services; developer works on the implementation of the software; integrator runs software verifying if it works and register packages; tester submits and stores test; release manager is responsible for defining the release candidate of the project, publishing packages, creating release notes and other documentation; guest has only read access.

2.2.1 Configuration service

The Configuration service is the component providing business logic for configuring software and running remote builds and tests. It is used by both CLI and WP.

The Configuration service abstracts the data storage backend, which holds the persistent version of the ETICS data model. It provides a set of high level methods to read and write objects from and into the database and a set of specialized methods either to generate and manipulate objects or to get information about the

[4]A local build or test is performed, for instance, by the developer on his/her personal workstation whenever he/she wants. A remote build or test is submitted, for instance, by the developer on a remote system that will process it when possible (i.e., the developer does not have a total control of the machines that may be used by other users).

operations performed by users and the system for auditing and logging purposes. In addition, the Configuration service abstracts the access to the underlying job [5] Execution Engine framework presenting a common build and test job interface layer for remote submission.

It is developed in Java and runs in a servlet container, such as Tomcat.

2.2.2 Repository service

The Repository service is the component providing logic for accessing build (or test) reports, artifacts and yum [17] repositories used by WP. It consists of a data management system that allows storing, cataloguing, browsing, searching, disseminating and deleting ETICS artifacts or sets of them.

All reports and artifacts produced by the ETICS system are stored in the repository. Its storage is divided in two parts: 1. a volatile storage area used by developers to access build and test artifacts that have been remotely built using the ETICS infrastructure, and 2. a permanent storage area where the official artifacts are registered typically by integrators. In both cases the built reports and artifacts are available for downloading.

It is developed in Java and runs in a servlet container such as Tomcat.

2.2.3 Execution engine

The execution engine allows the ETICS system to offer to the host projects the automation of remote builds and tests, possibly on a regular schedule and on a large set of different machines and platforms.

The engine is provided by the Metronome [21] build and test framework produced by the Condor Project (http://www.cs.wisc.edu/condor) that manages distributed job execution and provide access to computing resources. Metronome uses the ETICS CLI during the compiling and testing of software downloaded on a given machine.

[5]Job in ETICS is a list of instructions to be executed remotely to get the outputs.

3 DISSEMINATOR

3.1 Problem Description

Software development is facing the problem of improving the quality of software products. The lack of quality, in fact, can easily lead to major cost and delays in the development and maintenance of the software. One of the principal purposes of the ETICS project was to support software developers and managers in assuring the quality of their software. In order to achieve this goal ETICS users have been supplied with a graphical tool which enables them to control the behaviour of the values of some metrics over time with simple and clear charts.

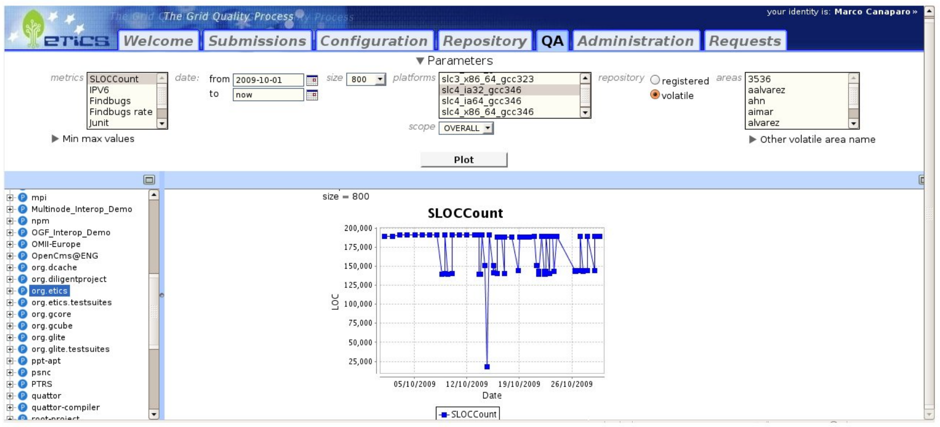
3.2 Solution

The ETICS Disseminator for trend-analysis is the portlet of the ETICS portal (shown in Figure 4) that has been designed and developed with the aim of providing ETICS users with a tool for monitoring the ”quality” of their code.

The ETICS Disseminator creates charts which represent the time variation of one or more metrics whose values have been previously collected by the ETICS system. The layout of the ETICS Disseminator is organized in three panels. The first one is a horizontal panel located on the top of the Web page which shows the form where it is possible to specify the metrics you want to be shown and some options to filter the results (i.e., date range, platforms’ name, type of repository - registered or volatile - and volatile area name). The second panel is a vertical one and is located on the left. It shows the expandable tree composed of all the projects with related subsystems, components and configurations registered in ETICS. Finally the third panel is on the right of the tree panel and it is filled with the graphs of the metrics, result of the request issued by the user.

3.2.1 Type of metrics

ETICS plugin framework is the feature of ETICS client that is related to the analysis and metric collection. Besides activating the existing plugins, ETICS users can leverage the extensibility of the framework adding their own ones. The existing



metrics: SLOCCount, IPV6, Findbugs, Findbugs rate, Junit
date: from 2009-10-01 to now
size: 800
platforms: slc3_x86_64_gcc323, slc4_ia32_gcc346, slc4_ia64_gcc346, slc4_x86_64_gcc346
scope: OVERALL
repository: registered [✓ volatile]
areas: 3536, aalvarez, ahn, aimar, alvarez
size = 800

Figure 4: A screenshot of the ETICS Disseminator.

plugins generate a variety of metrics which are listed, with a brief description, in Table 1. Some of these plugins, such as SLOCCount[6] and IPV6 [7], generate only one metric. The Findbugs [12, 13] and Junit [18] plugins create two metrics each. The CKJM [5] plugin provides eight metrics.

3.2.2 Metrics and charts

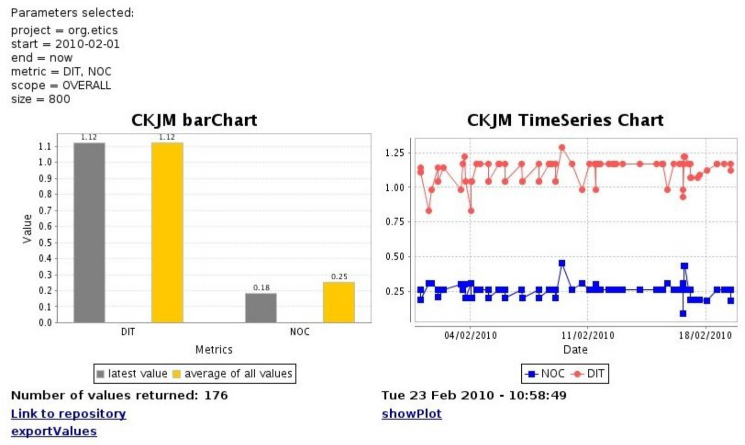
The ETICS Disseminator for trend-analysis is able to create one or two different images for each metric. Different types of charts have been chosen in order to give to the users the possibility to analyse the behaviour of the code they are writing. In particular, a time series chart is used for most of the metrics (SLOCCount, IPV6, Findbugs, Junit, and all the CKJM metrics), a pie chart has been considered a good choice for the metrics Findbugs rate and Junit rate, furthermore, a bar chart is plotted for all the metrics generated by the CKJM plugin. Pie charts and bar charts provide the users with the latest values obtained for a certain metric and the average of the other values for the same metric. Some examples of charts that the Disseminator create are shown below.

In Figure 5 a bar and a time series charts are shown for the metrics DIT and NOC for the `org.etics` project.

[6]Wheeler, D. A., SLOCCount, `http://www.dwheeler.com/sloccount`

Plugins	Metrics	Description
SLOCCount	SLOCCount	shows the number of lines of code.
IPV6	IPV6	represents the percentage of IPV6 compliance of code.
Findbugs	Findbugs	indicates the number of bugs found during the build or test.
	Findbugs rate	shows the percentage of modules that have successfully passed the threshold defined by ${findbugs.failure.threshold}.
Junit	Junit	shows the total number of tests successfully executed.
	Junit rate	indicates the percentage of components with tests successfully executed.
CKJM	WMC (Weighted Methods per Class)	the `ckjm` program assigns a complexity value of 1 to each method, and therefore the value of the WMC is equal to the number of methods in the class.
	DIT (Depth of Inheritance Tree)	provides for each class a measure of the inheritance levels from the object hierarchy top. In Java where all classes inherit Object the minimum value of DIT is 1.
	NOC (Number of Children)	measures the number of immediate descendants of the class.
	CBO (Coupling Between Object classes)	represents the number of classes coupled to a given class (efferent couplings). This coupling can occur through method calls, field accesses, inheritances, arguments, return types, and exceptions.
	RFC (Response For a Class)	measures the number of different methods that can be executed when an object of that class receives a message (when a method is invoked for that object).
	LCOM (Lack of Cohesion in Methods)	measures the number of different methods in a class that are not related through the sharing of some of the class's fields.
	Ca (afferent Couplings)	is a measure of how many other classes use the specific class.
	NMP (Number of Public Methods)	counts all the methods in a class that are declared as public. It can be used to measure the size of an API provided by a package.

Table 1: Metrics.


Parameters selected:
project = org.etics
start = 2010-02-01
end = now
metric = DIT, NOC
scope = OVERALL
size = 800

Number of values returned: 176
Link to repository
exportValues

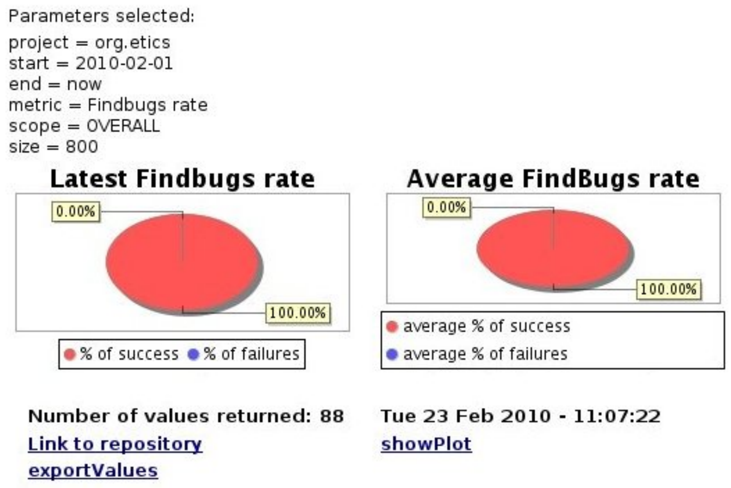
Tue 23 Feb 2010 - 10:58:49
showPlot

Figure 5: Two charts about the metrics DIT and NOC of the `org.etics` project. On the left a barchart is shown: for each metric there are a gray column and a yellow column that represent the latest value and the average value respectively. On the right there is a time series chart that shows the behaviour, over time, of the two metrics. Above the charts a summary of the parameters requested is shown.

In Figure 6 there are two pie charts that represent the latest and the average value of the metric Findbugs rate for the `org.etics` project. Finally in Figure 7 there are the plots of SLOCCount and Findbugs for the `org.etics.build-system` subsystem, starting from 2009-09-01 till now in the volatile repository.

3.2.3 Features of the charts

Together with the chart or the set of charts some information is provided. In particular, a summary of all the parameters chosen by the user is shown above the chart, whilst the overall number of the values returned with the date when the query was sent is shown below the chart. Moreover, some links are provided underneath these values: the first link is the ”link to repository” and redirects to the Web page of the ETICS Repository where all the results are shown in a table; the second one is called ”showPlot”: clicking on this link a new window is opened where only the charts and a small summary of the parameters chosen appear; and finally, the third link is named ”exportValues” and allows users to download, in


Parameters selected:
project = org.etics
start = 2010-02-01
end = now
metric = Findbugs rate
scope = OVERALL
size = 800

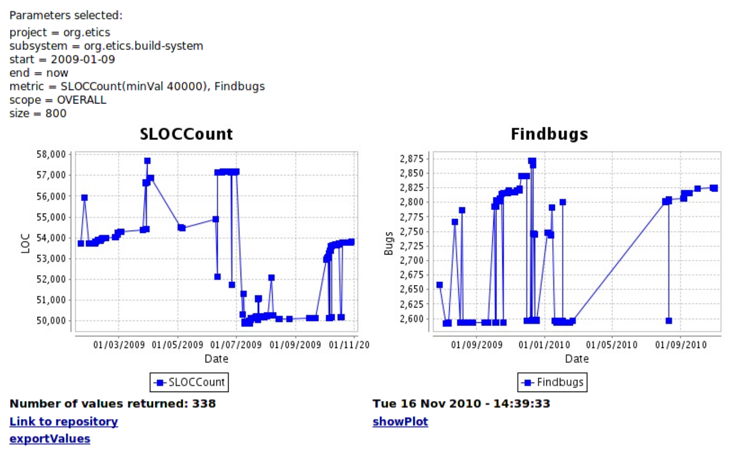
Number of values returned: 88
Tue 23 Feb 2010 - 11:07:22

Figure 6: Two pie charts. The left one represents the latest value of the metric Findbugs rate. The right one shows the average value of the same metric. The pie charts are red because both the percentage are 100% of success.

a csv (comma separated values) [24] file, all the results of the query for all the metrics with the date they have been stored in the ETICS Repository.

3.2.4 Integration with the ETICS Dashboard

The ETICS Disseminator for trend-analysis can be used by the ETICS Dashboard, which is a Web page composed of a variety of widgets that has the purpose of summarizing all the principal information about a project (for further details on the ETICS Dashboard see Section 4). The ETICS Disseminator may contribute to the information through the representation of some charts. Dashboard communicates with Disseminator via a specific URL. The ETICS Disseminator plots the graphs, saves them on some jpeg files and finally passes the binary files to the dashboard.



Parameters selected:
project = org.etics
subsystem = org.etics.build-system
start = 2009-01-09
end = now
metric = SLOCCount(minVal 40000), Findbugs
scope = OVERALL
size = 800

Number of values returned: 338
Link to repository
exportValues

Tue 16 Nov 2010 - 14:39:33
showPlot

Figure 7: SLOCCount and Findbugs graphs of the `org.etics.build-system` subsystem, the values are filtered with a time range starting from the 09th of January 2009 till now. On the left there is the chart related to the total number of lines of code, a minumum value of 40000 has been specified, on the right there is the number of bugs that Findbugs has found.

3.3 Implementation Details

3.3.1 How to make a query

The user has to specify at least one metric and one node of the tree (which can be a project, a subsystem, a component or a configuration). Optionally he or she can specify also a time range, a platform, a scope, a repository area and the name of a volatile repository area. All the parameters are passed from the client side of the GWT (Google Web Toolkit 2.0) [14] client application to the server side, where they are used to build a XPath [4] query that is sent to the ETICS Repository Web service. This Web service returns an XML file with all the values for the specified metrics. By parsing the file with JAXB (Java Architecture for XML bindings) [25] it is possible to organize the values according to the date and the name of the metric. Then all the values are plotted in the charts and saved in jpeg files. The images created, therefore, are shown in the plot panel of the disseminator. Figure 8 shows a sequence diagram of the process described. *EticsTrendAnalysis* is responsible for the creation of the portlet and it is the entrypoint of the application. *XPathQueryImpl* builds the XPath query that is sent to the ETICS Repository Web service and store all the results. Moreover, it invokes the makePlot function in order to create the jpeg files. *RepositoryWebService* is the ETICS Repository Web service that receive the XPath query and sends back all the results related to the parameters specified in the query. *XPathPlot* is responsible for the creation of the jpeg files and the plot of the charts.

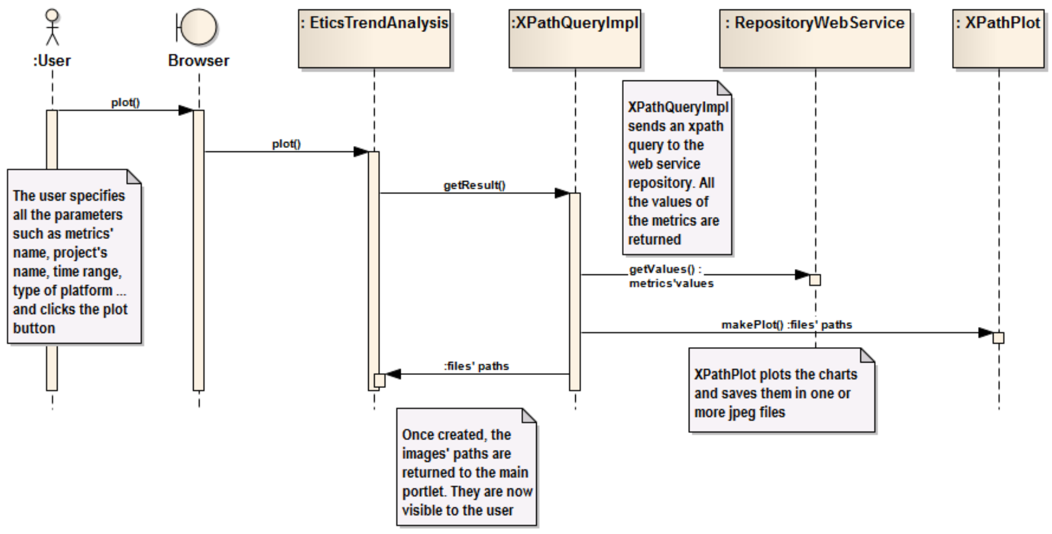

:User
Browser
: EticsTrendAnalysis
:XPathQueryImpl
: RepositoryWebService
: XPathPlot
plot()
plot()
getResult()
getValues() : metrics'values
makePlot() :files' paths
:files' paths
The user specifies all the parameters such as metrics' name, project's name, time range, type of platform ... and clicks the plot button
XPathQueryImpl sends an xpath query to the web service repository. All the values of the metrics are returned
XPathPlot plots the charts and saves them in one or more jpeg files
Once created, the images' paths are returned to the main portlet. They are now visible to the user

Figure 8: Sequence diagram of a request for one or more charts.

3.3.2 URL Query

As soon as the user has specified the parameters and clicked on the ”plot” button, all the options chosen appear in the URL bar of his browser. This mechanism allows the user to bookmark the page and to retrieve the same graphs with the updated data whenever desidered. Clicking on the bookmark, in fact, an URL query is sent to the ETICS Disseminator. Every parameter is parsed by the application and stored in one data structure. These data are processed in the same way as if they had been specified in the forms. Thus, the XPath query is sent to the Repository Web service and in a few seconds, the disseminator application receives the values from the Repository Web service and some charts will be plotted. The parameters that appear in the URL will be used to initialize the values of the forms of the disseminator and the tree on the left will be incrementally expanded with the purpose of selecting the projects, or subsystems or components chosen by the user.

3.3.3 Technologies

In the following the list of some of the most important technologies used by the ETICS Disseminator for trend-analysis is shown.

- **GWT** has been used for the ETICS Web portal. GWT is a development toolkit for building and optimizing complex browser-based applications.

- **Ajax** [9] (Asynchronous Javascript and XML) with Ajax the ETICS Disseminator can retrieve data from server asynchronously in the background with our interfering with the display and behaviour of the existing Web page. This technology has been used in order to retrieve all the various names of volatile areas, platforms, projects, subsystems, components and configurations in ETICS.

- **JAXB** allows Java developers to map Java classes to XML representations. The ETICS Disseminator for trend-analysis uses it to parse the XML file sent by the ETICS Web service and that contains all the values (with date) for all the metrics requested.

- **JFreeChart** [11] is an open-source framework for the programming language Java, which allows an easy creation of complex charts.

- **XPath** is a query language for selecting nodes from an XML document.

4 DASHBOARD

4.1 Problem description

The development of complex software needs to be supported by Web-based tools able to collect and represent information organized in different views [8]. A project manager, for instance, may need to analyse the state of some projects by evaluating their metrics and charts plotted on a single Web page. At the same time, a developer may be interested in tracing the quality of his or her code. One of the purposes of the ETICS 2 project was to provide a graphical tool that makes easily available a large set of information from different sources (like Web portal and Web repository), and shows measures of metrics about the quality and trend of the software projects.

4.2 Solution

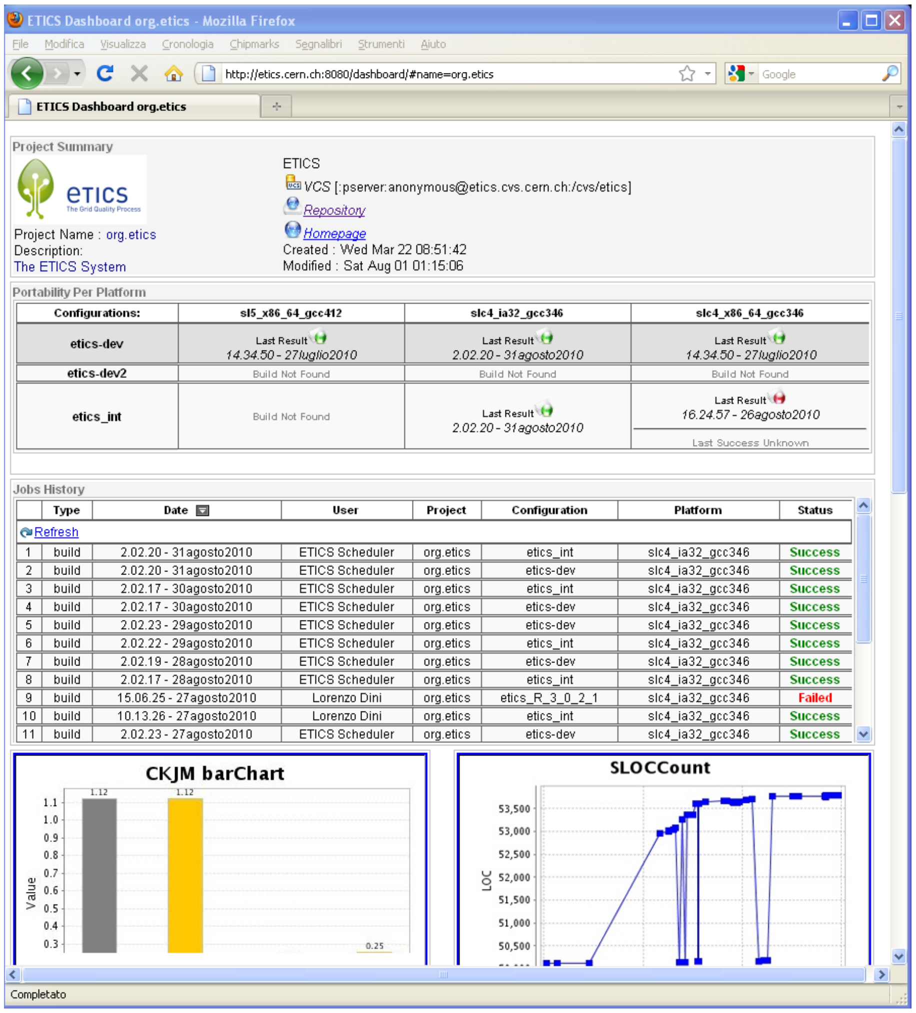
The ETICS Dashboard has been designed and developed with the aim of providing ETICS users with a tool for monitoring the builds and tests results, the quality of software and other useful project information. It contributes to an instant view of trends, patterns and anomalies in the software development activity.

The ETICS Dashboard Web page (shown in Figure 9) is a collection of widgets that offers to the ETICS users some different features. The Web page layout uses a grid composed of $\mathcal{M}$ *rows* and $\mathcal{N}$ *columns*, whose single cell is represented by the couple *(row, column)*. Each widget is placed in a cell. The features of each widget are represented by some key parameters: the location of the cell, which the widget is assigned to, given by the couple (row, column); and the span of the widget given by *rowSpan* and *columnSpan*. The values of the *rowSpan* and *columnSpan* parameters explain how many cells the widget occupies respectively vertically and horizontally.

An example of the ETICS Dashboard Web page organization is given in Figure 10(a) and Figure 10(b): the former shows 9 widget positions of the form (row, column); whilst the latter shows 9 widgets dimensions of the form (rowSpan, columnSpan).

4.2.1 Supported Widgets

The ETICS Dashboard provides a variety of widgets (listed in Table 2), whose description is detailed in the following paragraphs.


ETICS Dashboard org.etics - Mozilla Firefox
File Modifica Visualizza Cronologia Chipmarks Segnalibri Strumenti Aiuto
http://etics.cern.ch:8080/dashboard/#name=org.etics
Google
ETICS Dashboard org.etics
eTICS The Grid Quality Process
CKJM barChart
Value
SLOCCount
LOC
Completato

Project Summary

Project Name : org.etics
Description:
The ETICS System

ETICS
VCS [:pserver:anonymous@etics.cvs.cern.ch:/cvs/etics]
Repository
Homepage
Created : Wed Mar 22 08:51:42
Modified : Sat Aug 01 01:15:06

Portability Per Platform

Configurations:	sl5_x86_64_gcc412	slc4_ia32_gcc346	slc4_x86_64_gcc346
etics-dev	Last Result 14.34.50 - 27luglio2010	Last Result 2.02.20 - 31agosto2010	Last Result 14.34.50 - 27luglio2010
etics-dev2	Build Not Found	Build Not Found	Build Not Found
etics_int	Build Not Found	Last Result 2.02.20 - 31agosto2010	Last Result 16.24.57 - 26agosto2010 Last Success Unknown

Jobs History

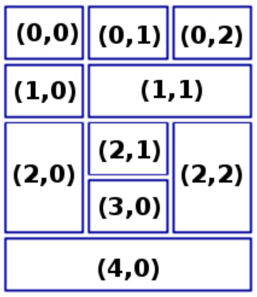
Refresh

	Type	Date	User	Project	Configuration	Platform	Status
1	build	2.02.20 - 31agosto2010	ETICS Scheduler	org.etics	etics_int	slc4_ia32_gcc346	Success
2	build	2.02.20 - 31agosto2010	ETICS Scheduler	org.etics	etics-dev	slc4_ia32_gcc346	Success
3	build	2.02.17 - 30agosto2010	ETICS Scheduler	org.etics	etics_int	slc4_ia32_gcc346	Success
4	build	2.02.17 - 30agosto2010	ETICS Scheduler	org.etics	etics-dev	slc4_ia32_gcc346	Success
5	build	2.02.23 - 29agosto2010	ETICS Scheduler	org.etics	etics-dev	slc4_ia32_gcc346	Success
6	build	2.02.22 - 29agosto2010	ETICS Scheduler	org.etics	etics_int	slc4_ia32_gcc346	Success
7	build	2.02.19 - 28agosto2010	ETICS Scheduler	org.etics	etics-dev	slc4_ia32_gcc346	Success
8	build	2.02.17 - 28agosto2010	ETICS Scheduler	org.etics	etics_int	slc4_ia32_gcc346	Success
9	build	15.06.25 - 27agosto2010	Lorenzo Dini	org.etics	etics_R_3_0_2_1	slc4_ia32_gcc346	Failed
10	build	10.13.26 - 27agosto2010	Lorenzo Dini	org.etics	etics_int	slc4_ia32_gcc346	Success
11	build	2.02.23 - 27agosto2010	ETICS Scheduler	org.etics	etics-dev	slc4_ia32_gcc346	Success

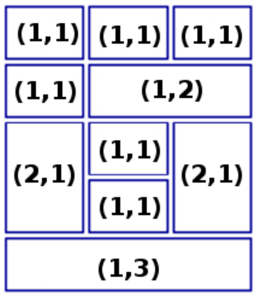
Figure 9: The screenshot of the ETICS Dashboard page.

ProjectSummary collects generic information about a given project such as repository URL, homepage URL, and logo.

Figure 11 shows an example of the ProjectSummary widget for the *org.etics* project. Besides the main project information, two Web links are available: the former (named Repository) redirects to the ETICS Repository Web page, the latter (named Homepage) to the Home Web page of the project.



(a) Widget position in terms of couples of form (row, column).



(b) Widget dimensions in terms of couples of form (rowSpan, columnSpan).

Figure 10: A representation of the ETICS Dashboard Web page layout.

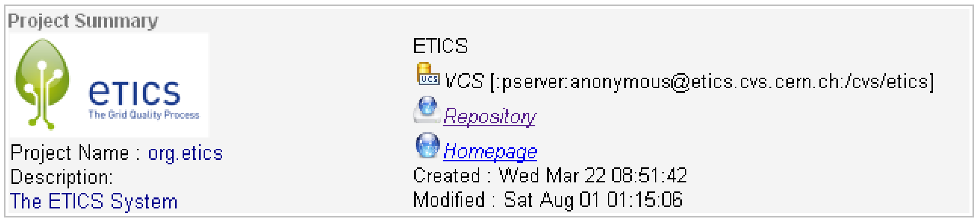
Widgets	Description
ProjectSummary	generic information about the project
JobHistory	build and test results over the time
Portability	build results per platform
Disseminator	charts plotted by the Disseminator
URL	Web page

Table 2: List of widgets

JobHistory lists the history of the submitted jobs to the ETICS infrastructure in a table. Each row contains the type of the submission (e.g., build or test) and its result. All records are displayed in discending order.

Figure 12 shows an example of the JobHistory widget for the *org.etics* project. The *Refresh* link allows user to reload all data from the Web services. Clicking a row, a panel with build or test job details (such as status, submission-id and links to repository and packages) is opened: the *View Reports* and *Browser Packages* links open, respectively, the report Web page and the repository Web page of the selected job.

Portability shows the state of the last build results per platform for a list of the ETICS configurations. The results are represented in a table in which rows


Project Summary

ETICS

VCS [:pserver:anonymous@etics.cvs.cern.ch:/cvs/etics]

Repository

Homepage

Project Name : org.etics

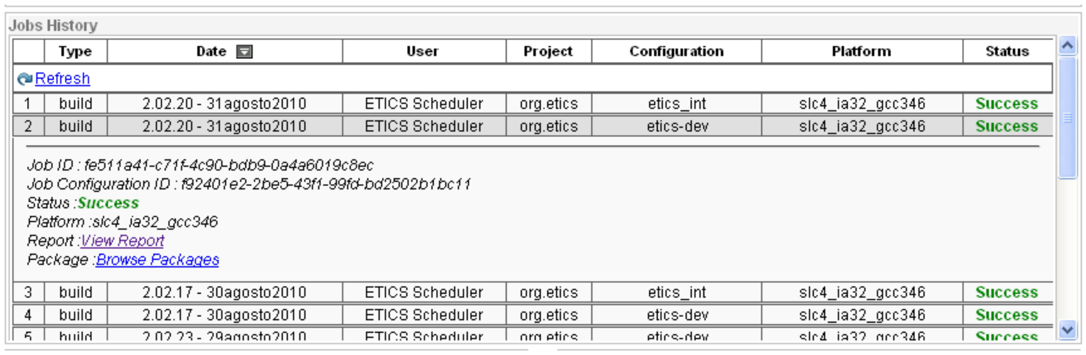
Description:

The ETICS System

Created : Wed Mar 22 08:51:42

Modified : Sat Aug 01 01:15:06

Figure 11: An example of ProjectSummary widget output for the *org.etics* project.


Jobs History

Refresh

	Type	Date	User	Project	Configuration	Platform	Status
1	build	2.02.20 - 31agosto2010	ETICS Scheduler	org.etics	etics_int	slc4_ia32_gcc346	Success
2	build	2.02.20 - 31agosto2010	ETICS Scheduler	org.etics	etics-dev	slc4_ia32_gcc346	Success

Job ID : fe511a41-c71f-4c90-bdb9-0a4a6019c8ec
Job Configuration ID : f92401e2-2be5-43f1-99fd-bd2502b1bc11
Status :Success
Platform :slc4_ia32_gcc346
Report :View Report
Package :Browse Packages

	Type	Date	User	Project	Configuration	Platform	Status
3	build	2.02.17 - 30agosto2010	ETICS Scheduler	org.etics	etics_int	slc4_ia32_gcc346	Success
4	build	2.02.17 - 30agosto2010	ETICS Scheduler	org.etics	etics-dev	slc4_ia32_gcc346	Success
5	build	2.02.23 - 29agosto2010	ETICS Scheduler	org.etics	etics-dev	slc4_ia32_gcc346	Success

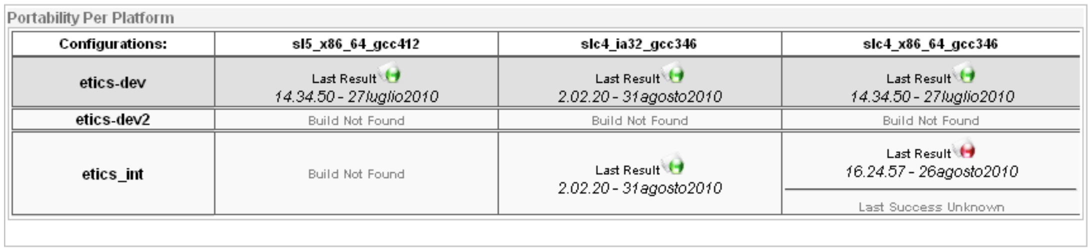
Figure 12: An example of Job History widget output.

are configurations and columns are platforms. In case of no results for the (configuration, platform) couple in the given time range, the message *"Build Not Found"* is shown. In case of failure a red icon appears next to the *"Last Result"* message and the last successful build is searched: if it is found, the *"Last Success"* message is shown; otherwise, the *Last Success Unknown* message is displayed.

Figure 13 shows an example of the Portability widget. Clicking on the green or red icons the details of the related Build Report from the ETICS repository is shown in a new Web page.

Disseminator makes available some different charts from the ETICS Disseminator component.

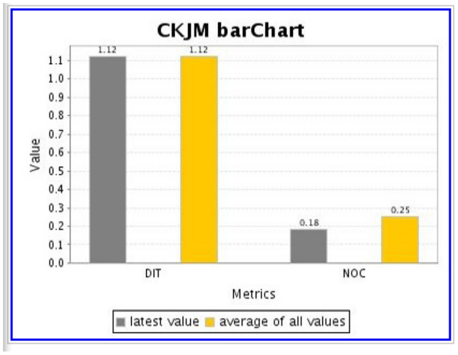
In the following, the CKJM bar chart (Figure 14(a)) and the *SLOCount* chart (Figure 14(b)) for the *org.etics* project are shown. The DIT and NOC metrics are plotted in the same CKJM chart.



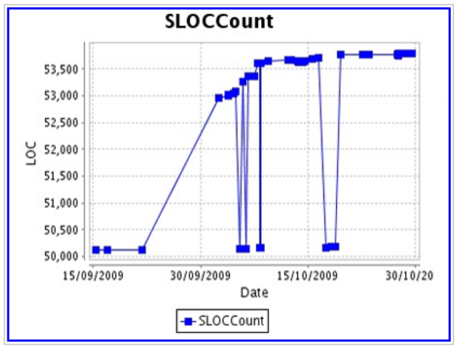
Portability Per Platform

Configurations:	sl5_x86_64_gcc412	slc4_ia32_gcc346	slc4_x86_64_gcc346
etics-dev	Last Result ✓ 14.34.50 - 27luglio2010	Last Result ✓ 2.02.20 - 31agosto2010	Last Result ✓ 14.34.50 - 27luglio2010
etics-dev2	Build Not Found	Build Not Found	Build Not Found
etics_int	Build Not Found	Last Result ✓ 2.02.20 - 31agosto2010	Last Result ✗ 16.24.57 - 26agosto2010 Last Success Unknown

Figure 13: An example of the Portability widget output.



(a) CKJM bar chart

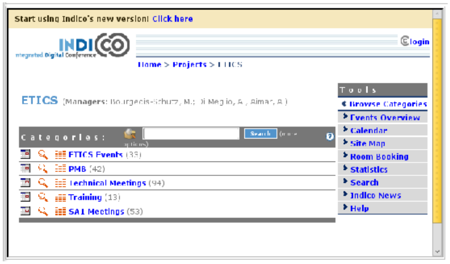


(b) SLOCCount chart

Figure 14: A couple of commonly used charts.

URL shows some useful URLs and Web page by using an *iFrame* HTML.

Figure 15(a) and Figure 15(b) show two examples of Web pages loaded by the URL widget.



Start using Indico's new version! Click here

INDICO integrated Digital Conference

login

Home > Projects > ETICS

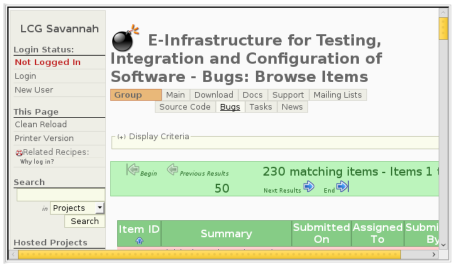
ETICS (Managers: Bourgeois-Schutz, M.; Di Meglio, A.; Aimar, A.)

Categories: Search (more options)

ETICS Events (33)
PMB (42)
Technical Meetings (94)
Training (13)
SA1 Meetings (53)

Tools
Browse Categories
Events Overview
Calendar
Site Map
Room Booking
Statistics
Search
Indico News
Help

(a) points to *http://indico.cern.ch*



LCG Savannah

Login Status:
Not Logged In
Login
New User

This Page
Clean Reload
Printer Version
Related Recipes:
Why log in?

Search
in Projects
Search

Hosted Projects

E-Infrastructure for Testing, Integration and Configuration of Software - Bugs: Browse Items

Group | Main | Download | Docs | Support | Mailing Lists
Source Code | Bugs | Tasks | News

(+) Display Criteria

Begin Previous Results 230 matching items - Items 1 t 50 Next Results End

Item ID	Summary	Submitted On	Assigned To	Submi By

(b) points to *https://savannah.cern.ch*

Figure 15: Some examples of URL Widget content.

4.3 Implementation Details

Two customization levels are required for the ETICS Dashboard page: the former is about the layout in relation to the position and the size of the widgets, whilst the latter concerns the type of the chosen widgets according to users' needs.

In order to implement both levels of customizations an XML-based configuration file has been introduced.

4.3.1 Configuration file overview

The configuration file of the ETICS Dashboard Web page is composed of two main sections: the former describes the page layout, whilst the latter includes the list of widgets to be loaded in the page.

Each configuration file has a unique name given by a specific key called *PAGE-ID*. The name format of the file is, for example, *etics.dashboard.conf.PAGE-ID.xml*. The *PAGE-ID* key is also used in the URL to point to the correct Dashboard Web page: the following URL

https://etics.cern.ch:8080/dashboard/#name=***MYPAGE***

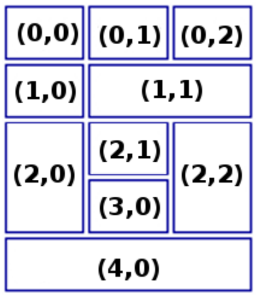
tells the Dashboard's server-side process to use the file named *etics.dashboard.-conf.MYPAGE.xml* to load all configuration parameters.

If no configuration file with the specified name is found, the Dashboard loads a default page, whose content depends on the default configuration file. On the contrary, the server-side servlet passes the file to another server-side process, called *DashboardPageLoader*, whose task is to load all data about the page layout and the widget parameters.

4.3.2 Dashboard Layout

The first section of the configuration file refers to the page layout. According to the description in Section 4.2, the following part of the configuration file specifies for the Dashboard page: the maximum layout width of 880 pixels; and the number of cells divided into 5 rows and 3 columns .

```
<layout>
  <params>
    <param>
      <name>numRows</name>
      <value>5</value>
    </param>
    <param>
      <name>numCols</name>
      <value>3</value>
    </param>
    <param>
      <name>maxWidth</name>
      <value>880</value>
    </param>
  </params>
</layout>
```



The grid of the ETICS Dashboard page.

4.3.3 Configuration of Widgets

The configuration file defines the list of widgets and the related parameters. For each widget, an XML tag, called `widget`, contains the complete list of parameters specified by the tag `param` name. Each parameter is composed of a `(name, value)` pair of tags. An extract of a configuration file shows the XML widget tag.

```
<widget>
   <params>
     <param>
       <name>type</name>
       <value>ProjectSummaryWidget</value>
     </param>
     <param>
       <name>project</name>
       <value>org.etics</value>
     </param>
     ...
   </params>
</widget>
```

An amount of common parameters is listed below:

type: takes one of the following: ProjectSummaryWidget, JobHistoryWidget, URLWidget, PortabilityWidget or DisseminatorWidget.

project, subsystem, component or **configuration**: are the ETICS name of project (like *org.etics*), subsystem (like *org.etics.portal*), component or configuration which widget refers to.

width and height: are the width and height values of the widget.

rowIdx and colIdx: are the row and column numbers that identify the cell where the widget is placed.

colSpan and rowSpan: are the row-span and column-span values for the cell of the widget. The default value is 1.

4.3.4 Widget interaction

The user can interact with any type of widgets. For example, when a row in the *JobHistory* widget is clicked, the browser generates a related *event*. The widget uses this event to call a server-side process that collects all the job details and then returns them to the client-side.

Figure 16 shows the sequence diagram that describes this behaviour.

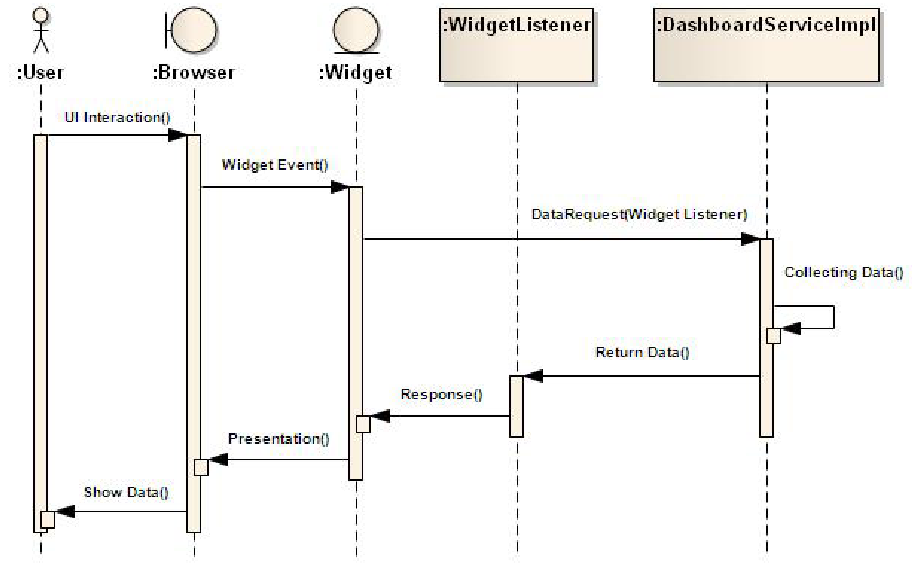


Figure 16: The sequence diagram for a typical interaction between the widget and the server-side application.

DashboardServiceImpl is the server-side class that receives and evaluates the call from *Widget*. It needs from *Widget* a *WidgetListener* that is used to return data back to *Widget*. At the end, *Widget* takes all the data from the *WidgetListener* and shows them in the proper way.

4.3.5 Technologies

The most important technologies used by the ETICS Dashboard are **GWT** and **JAXB** detailed in Section 3.3.3. The ETICS Dashboard uses JAXB to parse the XML configuration file, load and configure all widgets.

5 RELATED WORK

ETICS provides users with a variety of functionalities most of which are neglected by the subject of this paper, that is focused on the ETICS Dashboard and Disseminator. In the following paragraphs ETICS is compared to a couple of tools for software development and project management, mainly highlighting their dashboard features.

Jira provides issue tracking and project tracking for software development teams to improve code quality and the speed of development (`http://www.atlassian.com/software/jira/`). Both Jira's and ETICS's dashboard aim at providing a Web Page where users can get the basic information about their products at a glance. Another similarity between Jira and ETICS is the fact that both have been designed as aggregator of self-content elements that allows users to promptly and quickly visualize information. Jira calls these elements gadgets, whereas ETICS widgets. The ETICS widgets are a subset of the Jira's gadgets: adding Web Page, graphics and charts, report and summary of products information. However, widgets support all the basic functionalities provided by Jira. ETICS, as Jira, is able to define different Dashboard configurations according to users' needs, providing custom monitoring information. ETICS, unlike Jira, is more flexible in the maximum number of widgets in each dashboard configuration.

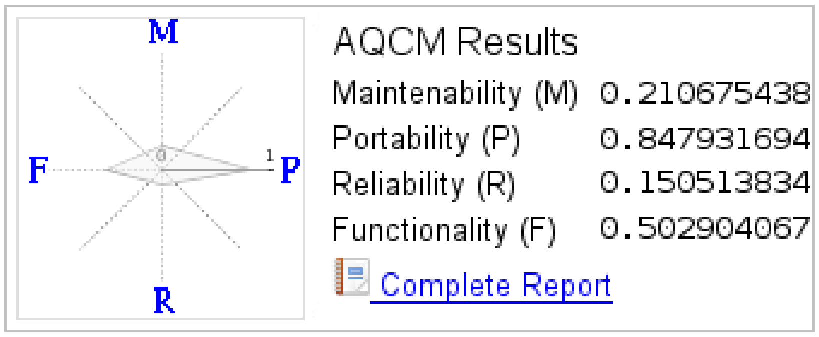
Redmine (`http://www.redmine.org/`), another tool for project management, provides users with a plugin, called Scrumdashboard, in order to visualize Versions, Tracks and Issues of a given project. Its use is mainly focused on the showing of patches states, opened tracks and issues. The ETICS Dashboard is extremely flexible being founded on a set of configurable widgets: therefore it might be easily extended to provide functionalities similar to Redmine's, like a widget for tracking.

6 FUTURE WORK

The ETICS Disseminator for trend-analysis is able to plot further charts that show the metrics of the AQCM (Automated Quality Certification Model) plugin [22]. This plugin supports different programming languages (Java, C/C++, Python) by exploiting results coming from different plugins - such as PMD, CKJM, Checkstyle, CCCC, CCN, SlocCount, IPv6, Codewizard, PyCyCom - in order to provide 4 more metrics: maintainability, reliability, portability, functionality. These metrics can measure the compliance of the software to several ISO/IEC 9126 aspects. More in detail, the maintainability metric measures the ability of the software to be modified with relatively little effort, the reliability metric provide the user with the ability to keep the agreed performances within the agreed conditions. The portability metric measures how the software can be moved from an operating system to another one, finally, the functionality metric represents the ability provide functions which meet the explicit and the implicit functional requirements of the software.

The ETICS Disseminator for trend-analysis can plot time series charts of these four metrics in order to let the users know what is the behaviour of one or more of the AQCM metrics over time. Moreover, if the user specifies all four metrics a polar chart about the latest values will be shown. As soon as some values related to the AQCM metrics are collected by the ETICS system, users will be able to exploit the further information given by the AQCM charts in order to evaluate their code.

Concerning the ETICS Dashboard users can simply customize the Disseminator widget in the Dashboard configuration file with the AQCM parameters. The following Figure 17 shows a possible representation of the AQCM chart in the AQCM widget.



AQCM Results

Maintenability (M)	0.210675438
Portability (P)	0.847931694
Reliability (R)	0.150513834
Functionality (F)	0.502904067

Figure 17: A possible representation of the AQCM chart for the AQCM widget. In the left side it is placed the AQCM polar chart retrieved from the ETICS Disseminator, whilst in the right it is reported the summary of the values collected and the link to the complete AQCM ETICS plugin report.

7 CONCLUSIONS

This paper described a data environment for software development process developed as part of the software quality program of the ETICS 2 European project. This data environment was built in response to some of the requirements coming from the ETICS users. It integrates two main aspects: 1) the trend analysis disseminator; and 2) the representation of quantitative data (i.e., metrics) according to a standard and centralized organizational dashboard. The set of metrics taken into consideration are restricted but significant, however new metrics can be included implementing new plugins. This data environment has been put in production since August 2010.

We are confident that its use will enable the software team to improve the software process. In addition, this data environment will enforce the use of standards that guarantee that all metrics are comparable; it will reduce the workload and increase the consistency and regularity of data collection; it will provide analytical instruments that improve the communication of metrics results to all members of the software team.

8 ACKNOWLEDGEMENTS

This work was done in the context of the ETICS 2 project funded by the European Commission under contract number INFSO-RI-223782.

References

[1] B. Allcock, I. Foster, V. Nefedova, A. Chervenak, E. Deelman, C. Kesselman, J. Lee, A. Sim, A. Shoshani, B. Drach, and D. Williams. High-performance remote access to climate simulation data: A challenge problem for data grid technologies. In *Proceedings of the SC2001 Conference*, Denver, November 2001.

[2] M.-E. Begin, S. Da Ronco, G. Diez-Andino Sancho, M. Gentilini, E. Ronchieri, and M. Selmi. Etics meta-data software editing - from check out to commit operations. *Journal of Physics: Conference Series*, 119, 2008.

[3] M.-E. Begin, G. Diez-Andino Sancho, A. Di Meglio, E. Ferro, E. Ronchieri, M. Semli, and M. Zurek. Build, configuration, integration and testing tools for large software projects: Etics. *Lecture Notes in Computer Science (LNCS)*, 4401/2007(4401):81–97, 2007. N.Guelfi and D. Buchs (Eds.): RISE 2006.

[4] A. Berglund, S. Boag, D. Chamberlin, M. F. Fernandez, M. Kay, J. Robie, and J. Simon. Xml path language (xpath) 2.0 w3c recommendation, January 2007.

[5] S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Transaction on Software Engineering*, 20(6):476–493, 1994.

[6] R.G. Cooper. *Winning at New Products: Accelerating the Process from Idea to Launch, Third Edition [Paperback]*. Basic Books; 3rd edition, June 2001.

[7] S. Deering and R. Hinden. Internet protocol, version 6 (ipv6) specification, Decembre 1998.

[8] S. Few. *Information Dashboard Design: The Effective Visual Communication of Data*. O'Reilly Media, 1 edition (January 1, 2006).

[9] J. J. Garrett. Ajax: A new approach to web applications, February 2005.

[10] A. Giesler, A. Streit, E. Ronchieri, and M. Dibenedetto. Demo: Synergizing etics and unicore software. In *UNICORE Submit 2010*, pages 37–43, Julich, Germany, 18-19 May 2010.

[11] D. Gilbert. The jfreechart class library, June 2002.

[12] C. Grindstaff. Findbugs, part 1: Improve the quality of your code. why and how to use findbugs, May 2004.

[13] C. Grindstaff. Findbugs, part 2: Writing custom detectors. how to write custom detectors to find application-specific problems, May 2004.

[14] R. Hanson and A. Tacy. *GWT IN ACTION*. Manning, 2007.

[15] W. Harrison. A universal metrics repository. In *Pacific Northwest Software Quality Conference*, Portland, Oregon, USA, October 18-19 2000.

[16] Ford W. Polk W. Housley, R. and D. Solo. *Internet X.509 Public Key Infrastructure - Certificate and CRL Profile*, January 1999.

[17] M. Jang. *Linux Patch Management: Keeping Linux Systems Up To Date*, volume Chapter 6. Prentice Hall. Part of the Bruce Perens' Open Source Series series., Jan 9 2006.

[18] A. J. S. Mills. *JUnit Testing Utility Tutorial*. The University Of Birmingham, 2005.

[19] E. Olson, R. Walker, and Ruekert R. Organizing for effective new product development: the moderating role of product innovativeness. *Marketing*, 59:48–62, 1995.

[20] E. Palza, C. Fuhrman, and A. Abran. Establishing a generic and multidimensional measurement repository in cmmi context. In *28th Annual NASA Soft. Eng. Workshop (SEW'03)*, pages 12–20, 2003.

[21] A. Pavlo, P. Couvares, R. Gietzel, A. Karp, I. D. Alderman, Livny M., and C. Bacon. The nmi build & test laboratory: Continuous integration framework for distributes computing software. In *the 20th conference on Large Installation System Administration, Washington, DC*, pages 263–273, December 2006.

[22] Takacs E. Matranga I. Di Meglio A. Rippa, A. and A. Manieri. Etics system: Automated testing and quality assurance. In *QA&TEST 2009, 8th International Conference on Software QA and Testing on Embedded Systems*, 2009.

[23] J. Ronkainen, T. Rahikkala, and R. Blackwood. Automating scm metric data collection and analysis in virtual software corporations. In *Proceedings of 25th EUROMICRO Conference*, volume 2, pages 279 – 283, Milan, Italy, 08 set 1999 – 10 set 1999 1999.

[24] Y. Shafranovich. Common format and mime type for comma-separated values (csv) files, October 2005.

[25] D.-O. Simion. Java facilities in processing xml files - jaxb and generating pdf reports. *Informatica Economica*, 2008.

[26] V. Subramanyam and S. V. B. Sharma. Hpd - query tool on projects historical database. In *SEPG Conference*, Bangalore, February 1999.

[27] C. Szyperski. *Component Software: Beyond Object-Oriented Programming. 2nd ed.* Addison-Wesley Professional, Boston, 2002.

[28] L. Westfall. 12 steps to useful software metrics. Technical report, The Westfall Team, PMB 101, 3000 Custer Road, Suite 270, Plano, TX 75075, 2005.