Project Acronym	***BlueBRIDGE***
Project Title	***Building Research environments for fostering Innovation, Decision making, Governance and Education to support Blue growth***
Project Number	***675680***
Deliverable Title	***Software Release Activity: Interim Report***
Deliverable No.	***D4.2***
Delivery Date	***October 2016***
Authors	***Gabriele Giammatteo, Maria Antonietta Di Girolamo***

BlueBRIDGE receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 675680

DOCUMENT INFORMATION

PROJECT	
Project Acronym	BlueBRIDGE
Project Title	Building Research environments for fostering Innovation, Decision making, Governance and Education to support Blue growth
Project Start	1st September 2015
Project Duration	30 months
Funding	H2020-EINFRA-2014-2015/H2020-EINFRA-2015-1
Grant Agreement No.	675680
DOCUMENT	
Deliverable No.	D4.2
Deliverable Title	Software Release Activity: Interim Report
Contractual Delivery Date	October 2016
Actual Delivery Date	November 2016
Author(s)	Gabriele Giammatteo (ENG), Maria Di Girolamo (ENG)
Editor(s)	Gabriele Giammatteo (ENG), Maria Di Girolamo (ENG)
Reviewer(s)	George Kakaletris (UOA)
Contributor(s)	n/a
Work Package No.	WP4
Work Package Title	VRE Deployment and Operation
Work Package Leader	ENG
Work Package Participants	CNR, ENG, UOA
Distribution	Public
Nature	Other
Version / Revision	1.0
Draft / Final	Final
Total No. Pages (including cover)	23
Keywords	Software integration, packages, distribution, release, testing

DISCLAIMER

BlueBRIDGE (675680) is a Research and Innovation Action (RIA) co-funded by the European Commission under the Horizon 2020 research and innovation programme

The goal of BlueBRIDGE*, Building Research environments for fostering Innovation, Decision making, Governance and Education to support Blue growth*, is to support capacity building in interdisciplinary research communities actively involved in increasing the scientific knowledge of the marine environment, its living resources, and its economy with the aim of providing a better ground for informed advice to competent authorities and to enlarge the spectrum of growth opportunities as addressed by the Blue Growth societal challenge.

This document contains information on BlueBRIDGE core activities, findings and outcomes and it may also contain contributions from distinguished experts who contribute as BlueBRIDGE Board members. Any reference to content in this document should clearly indicate the authors, source, organisation and publication date.

The document has been produced with the funding of the European Commission. The content of this publication is the sole responsibility of the BlueBRIDGE Consortium and its experts, and it cannot be considered to reflect the views of the European Commission. The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated the creation and publication of this document hold any sort of responsibility that might occur as a result of using its content.

The European Union (EU) was established in accordance with the Treaty on the European Union (Maastricht). There are currently 27 member states of the European Union. It is based on the European Communities and the member states' cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice, and the Court of Auditors (http://europa.eu.int/).

The information contained in this document represents the views of the BlueBRIDGE Consortium as of the date they are published. The BlueBRIDGE Consortium does not guarantee that any information contained herein is error-free, or up to date. THE BlueBRIDGE CONSORTIUM MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, BY PUBLISHING THIS DOCUMENT.

GLOSSARY

ABBREVIATION	DEFINITION
BlueBRIDGE	Building Research environments for fostering Innovation, Decision making, Governance and Education to support Blue growth
iMarine	Data e-Infrastructure Initiative for Fisheries Management and Conservation of Marine Living Resources
gCF	gCore Framework
ETICS	E-infrastructure for Test, Integration and Configuration of Software
E-IIS	Engineering Informatica Informatica S.p.A.
FAO	Food and Agriculture Organization of the United Nations
NKUA	National and Kapodistrian University of Athens
CNR	Consiglio Nazionale delle Ricerche
D4Science	Data Infrastructure for Science
BTRT	Build and Test Report Tool
SVN	Subversion
DT	Deployment Testing
DT Issue	Deployment Testing Issue
FT Issue	Functional Testing Issue
FT	Functional Testing
Tested on Preprod	The developer and/or the portal manager or the preprod infrastructure manager are confident that the component behaves as expected and can be released as is.

TABLE OF CONTENT

DELIVERABLE SUMMARY

The intent of this report is to describe the activities performed and the results achieved in the task T4.4 of BlueBRIDGE project in the reporting period (from M1 to M15). The first part of the document focuses on the updates done to the procedures and tools used in the task. The second part of the document lists the gCube releases rolled-out during the reporting period providing details and statistics about release content, meaningful events and milestones, integration and testing activities.

This document complements and updates deliverable D4.1 “Software Release Procedure and Tools”. In addition, the content of this deliverable is also available and further detailed in the gCube Wiki at:

https://wiki.gcube-system.org/gcube/Software_Integration_and_Distribution:_Overview

EXECUTIVE SUMMARY

Deliverable D4.2 – "Software Release Activity: Interim Report" provides the main outcome of the release activities performed in the period. They include BlueBRIDGE project wiki plus a set of documentation pages for single facilities hosted by the gCube wiki, the updating of the procedures and tools used within the Release Cycle and Maintenance Release Cycle Procedures (procedures introduced and described in the deliverable D4.1 – Software Release Procedure and Tools), the analysis of an alternative Software for the software integration and the introduction of a new storage repository for the source code and distribution of the software release.

In particular, this report provides the main activities performed for the:

- updating of the functional testing procedures with the introduction of a new figure: Testing Team, the new Test Plan (consist of a new Functional Master Table shared with all actors involved as developers, testers), the new template used by developers to describe the portlet test;
- introducing of the integration procedure to have a soft migration (for the end of the BlueBRIDGE project, on September 2018) from the release based on gCore components to SmartGears components (and consequently to cancel obsolete packages or software components);
- creation of the new guideline for the generation of Software distribution packages;
- the introduction of a new tools for the publication of the Software (GitHub), study of the possibility to adopt a popular Software for the continuous integration activities (Jenkins).

Finally, a detailed report is provided about the functional test at the end of this reports included the reports and statistics about the software release delivered on this period report (from M1 to M15).
This document complements and updates deliverable D4.1 "Software Release Procedure and Tools". In addition, the content of this deliverable is also available and further detailed in the gCube Wiki at:
https://wiki.gcube-system.org/gcube/Software_Integration_and_Distribution:_Overview

1 INTRODUCTION

The document is a report that describes all activities performed in task T4.4 of the BlueBRIDGE project in the first reporting period (M1 to M15).

The task T4.4 is in charge of managing the release process of the software produced in the project. In particular, this task (i) defines release procedures, (ii) establishes the release plan; (iii) coordinates the release process; (iv) operates the tools required to support the release activities; (v) keeps the software under continuous integration; (vi) performs functional validation of released web components (i.e. portlets); (vii) validates the software documentation; (viii) takes care of the distribution of the software. Procedures and the tools are continuously reviewed and improved in order to kept them aligned with project requirements, enhance the efficiency and to improve the quality of software delivered.

The activities described in the document are organized in four main sections:

Release Procedures Updates (cf. Sec. 2) that collects all the updates to procedures done in the reporting period. It includes the updates to the Functional Testing procedure, the Integration procedure (i.e. split of gCore and SmartGears releases, clean-up of obsolete components) and the new guidelines for the generation of distribution packages;

Release Tools Updates (cf. Sec. 3) that contains the upgrades to the tools used during the release procedure. It includes improvements to the ETICS tool, the publication on GitHub and the feasibility study for the adoption of Jenkins in continuous integration activities;

Testing Results (cf. Sec. 4) includes reports and statistics on the testing results for gCube 4.0.0 (the first release tested with the new procedure);

Release History (cf. Sec. 5) includes reports and statistics on the seven software releases (from gCube 3.9.0 to gCube 4.1.1) delivered by the project.

2 PROCEDURE UPDATE

The section describes the changes of the release procedures adopted, in order to improve:

- their efficiency and effectiveness,
- the quality of delivered software.

2.1 PORTLETS FUNCTIONAL TESTING

The Portlets Functional Testing procedure adopted in the BlueBRIDGE project, is inspired by the relevant procedure in the iMarine project, described in report "D7.4_Software_Release_Activity_Report".

The major changes with respect to the previous procedure are:

- the fields for the Functional Master Table as updated in the following way:
    - replaced the Domain Expert field with the Tester name field (the figure of the Domain Expert, now, coincides with the figure of the Owner);
    - replaced Notes and Functional Tests with the Link Test Plan (Notes filed now is provided in the Test Plan).
- new issue management for the deployment (DT issue, Deployed on Pre-prod) and functional testing (FT issue, Tested on pre-prod);
- new procedures created for the functional test portlets:
    - sharing of summary testing results in the wiki page (Functional Master Table);
    - definition of the new test plan in the VRE workspace, based on template provided by the Portal Manager.

The revision of the functional testing procedure assures that all gCube services meet the quality requirements:

- to define a system free from code anomalies, coding errors and erroneous operations;
- to respond to the design and architectural specifics of the gCube software;
- to be user-friendly;
- to give a proper (and rapid) response in case of wrong usage or defects that cannot entirely be eliminated;
- to streamline updating the infrastructure environment;
- to establish and introduce the Tester team.

The gCube software testing is segmented by typology of components:

- services and libraries are tested through the Deployment Test (https://wiki.gcube-system.org/gcube/Deployment_Testing);

- portlets are tested through the Functional testing. Additionally, during this test, the services and libraries are indirectly tested because the portlets depend on them to deliver their functionality. The new actors involved in this activity are: **Portal Manager**, **Infrastructure Manager**, **Release Manager**, **Deployer manager**, **Developers** and **Testers** team. The Tester Team is assembled in order to improve and facilitate the testing activities and include one/more person from each partner involved in Work Package 4. Each portlet artefact which is released must participate the functional testing procedure. Furthermore, if anything in the dependency chain of a Portlet X is released, then X has to be functionally tested too. The first step is to analyse what are the software artefacts provided by release and their software dependencies. The portal manager executes manually this task.

The Portlets Functional Testing consists of four stages:

1. WebArchive check,
2. Back end Service check,
3. Rendering check and
4. Functional Test.

The first three stages (Web Archive check, Back End Service check and Rendering check) are unchanged w.r.t. iMarine. (their definition and scope are reported in the iMarine project report D7.4 Software Release Activity Report).

The fourth stage is enhanced and its changes are about:

- the approach to share the results of the tests,
- the introduction of new software Test Plan procedure,
- the introduction of new fields for the Functional Test Master Table,
- the preparatory steps.

The details about each stage are hosted on the wiki page Portlet Functional Testing.

The new procedure for the Functional Testing regarding:

- Preparatory steps that involved Portal Manager and Release Manager;
- Roles for the Tester and Developer;
- Infrastructure Test Environment.

Preparatory steps

The preparatory steps involve the Portal and Release Manager. The details about the role of the Portal Manager, the Release Manager, the Software Testing Plan are hosted on the Functional Test Procedure wiki page, too.

The Software Testing Plan (#1413) is created in the gCube VRE Folders containing two folders: **Material** and **Releases**.

The **Material** directory provides the following information:

1. The general XLS template to be filled in by each portlet developer for compiling the Testing Plan and

2. the portlet folder (created by **Portal Manager**) for each portlet to be functionality tested.

The syntax used to create the portlet folder is: ***$portlet_name_folder***.

Every developer creates the actual testing plan into ***Material/$portlet_name_folder*** including by adding additional files required for the test (e.g. cvs_files). To describe the functional tests, the developer uses the Portlet Testing Plan Template ; this file is pasted and completed into the Release folders by the testers. A Task (Redmine Ticket) is assigned, by the **Release Manager**, to each portlet developer. The **Releases** directory contains a folder for each gCube release containing the tests to be executed. The Portal Manager creates (in this directory) the gCube release directory and the portlets directories related. The **Tester** can start with the functional test when Deployer/Infrastructure Manager updates the CRT: *{status: Under Integration -> Deployed on Preprod}*. The following steps are executed:

1. The Tester copies the TestPlan the Releases/gCube_release/portltes_directory to test. For example: the tester tests the invites-friends.1-1-0 portlets. The Tester copies the TestPlan for this portlets into Releases/org.gcube.4.0.0/invites-friends.1-1-0.

2. The Tester updates the TestPlan, provided in the Releases directory, with the results for the functional testing. For example, in the case of the invites-friends, the Tester updates the TestPlan with the functional testing results for this component.

In case the functional testing failed, the Tester opens the issue in the following way:

- The tester updates the CRT: *{status: Deployed on Preprod -> FT Issue, comment: <issue description>}*. The comment should always include a link to a relevant log/report file. The Tester updates the TestPlan providing the number of the issue ticket (field "Link to the Issue") and describing the test results (field "Results"), too.

- After the resolution of the issue, the developer updates the CRT: *{status: FT Issue -> Under Integration, comment: <issue description>}*. In this way, the Infrastructure Manager can re-deploy the service and update the CRT status: *{status: Under Integration -> Deployed on Preprod, comment: <issue description>}*. The tester knows that can repeat the testing and if the component passes the test, updates the CRT: *{status: Deployed On Preprod -> Tested on Preprod, comment: <issue description>}*. Otherwise one repeats the step 1) (the status return in FT Issue). The Tester updates the TestPlan with the new test results (field "Results"), too.

The following figure shows the flow of the components from “under integration” to “released”.

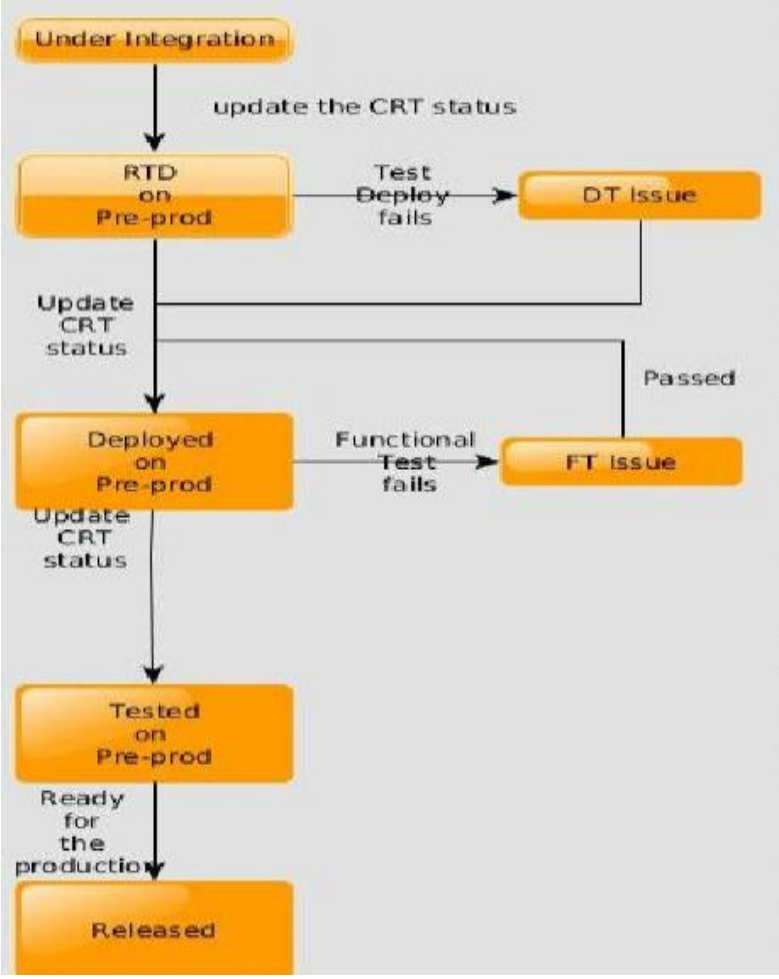

Figure 1. CRT State Diagram

The **pre-production infrastructure** for the functional testing is hosted at CNR.

Another responsibility for the Release Manager is to ask to **every partner** to suggest the persons that participate in the testing team. The following **rules** apply:

1. The **effort** should be based not per application but rather per **number of tests to be executed** (#1413).
2. The tester should **not** be the **same owner of the portlet (#1413)**.
3. For every release a wiki page is created: **$functional_test_master_table_Org_gCube_#Release**. For example for the release org.gcube.4.0.0 the following page has been created: Functional Test Master Table gCube Release 4.0.0. The Functional Test Master Table Template is filled by the testers with the FT results for the portlets applications in the pre-production infrastructure.

Role for the Tester

The details about the Tester's role are hosted on the Testers wiki page session.

The Tester is responsible:

- for the execution of the functionality tests and to submit functionality test bugs

- for filling in the dedicated Portlet Testing Plan Template and the Functional Test Master Table Template as described in the Functional Test (FT) Procedure.
- tracking the FT Issue through the use of the track system issue tool (ref. D4Science support).

The tester starts with the functional test when Deployer/Infrastructure Manager updates the CRT status: *{status: Under Integration -> Deployed on Preprod}.*

At the moment of the writing of this report two Functional Master Tables are provided, the list of these tables are hosted on the Functional Master Table List wiki page.

Role for the Developer

The details about the role of the Developer are described in the Developer Roles wiki page. The responsibility of the developer for the testing is :

1. to fix functionality test bugs
2. to fill the dedicated Portlet Testing Plan Template as described in the Functional Testing Procedure wiki page.

2.2 GENERATION OF DISTRIBUTION PACKAGE

Since the publication of gCube software on GitHub started in BlueBRIDGE project, as reported in the GitHub section of this report [18], a review of the procedure for the generation of distribution packages has been adopt to assure the increased quality of these packages.

The *distro* directory of gCube Maven components contains a set of files and templates used to generate files that will be included in the distribution packages of the component.

The following sections describes the mandatory and optional files expected to be in this directory. However, *distro* directory can also contain other component-specific files required to be distributed in the component packages.

The layout, the files and the filtering mechanism presented in this section work only for Maven components. They will not work with the **Ant** components. However, since also Ant components must provide the mandatory files and since Ant does not provide any support for filtering of files, developers must take care of keep the files updated manually.

The main changes regarding:

- the new mandatory files as: a) README, b) LICENSE, c) changelog.xml, d) profile.xml and e) descriptor.xml files.
- the deprecated files: the svnpath.txt file has been deprecated since the same information is automatically retrieved from component checkout commands;
- the new mechanism to find the Maven information for the software components. It provides from the variable-substitution mechanism using data found in Maven pom.xml.

The details for every file are available on the wiki page: Maven Distro Directory Layout.

2.3 INTEGRATION PROCEDURE UPDATES

The main changes in the Build procedure regarding:

- the splitting of releases and
- the re-factoring of the components.

Splitting Release

To simplify the migration from the gCore to SmartGear enabling technology, starting from the gCube 3.11.0 release, it has been decided to maintain and integrate in parallel two gCube configurations for each release: one based on gCore and one based on SmartGears.

The split of gCube in two different releases originated from a technical issue: new functionalities in common libraries needed to be released for SmartGear that were not compatible gCore. Since some gcore-based components used the same libraries, having two project configurations, allowed to have both the old and the new version of such libraries in the same release.

It became a convenient way to identify old/legacy components and "isolate" them in gCube 3.11.0-gcore, ready to be dismissed in the future.

Only components with a direct or indirect dependency on gCore (gCf, legacy-ws-core) went in gCube release-gcore. Portal, portlets, SmartGears-based, fws-based components went in gCube smartgear release. This splitting was needed to release some new functionalities in common libraries for SmartGears. Two project releases are configured on ETICS:

- org.gcube.3-11-0-gcore: with gCore and all components that depended on it;
- org.gcube.3-11-0: with all the other components.

This activity is managed through the use of the gCube's Redmine-based Issue Tracker. For the system tracking, the rule to create one ticket for each configuration released (project, subsystem, components) remains. Therefore, two different release tickets for gCube 3.11.0 and gCube 3.11.0-gcore, two different tickets for distribution-5.3.0-gcore and distribution-5.3.0, with the appropriate parent task are needed.
By default, developers should release their components in org.gcube-3.11.0 unless they knew that their components depended on gCore directly or indirectly.

Re-factoring gCube components

From the first month of the project component task is to clean-up and re-factor of the gCube components. The scope is to remove the obsolete components from the release. This activity continues until the end of the project, for every release.

Deployment Phase of Artifacts

The maven-builder is updated to simplify the deployment phase of artifacts and to generate new version numbers that include also SVN revision and timestamp. The details are hosted on Maven Builder wiki page session.

3 RELEASE TOOLS

This section describes the changes introduced in the tools used during the release procedure. All the changes have been applied either to improve the automation and efficacy of the tools and or to meet changes in the release procedure.

3.1 GITHUB

One of the planned activities for task T4.4 is the publication of gCube source code on GitHub platform. It is one of the major platforms for source code hosting and very popular in the open source community.

A repository for gCube software has been created on GitHub (https://github.com/gcube-system/gcube-releases), at the end of each release (starting from gCube 3.10.0):

- a source package is created for each component. Source packages are created after pom versions have been updated and distro files have been interpolated, see section D42_BlueBRIDGE_Software_Release_Activity_Interim_Report#Generation-of-Distribution-Packages. Maven components use assembly plugin to create the package (Ant components use and ad-hoc bash script to create the package);
- source packages are extracted in the correct location on the gCube Git repository;
- changes are committed and then pushed to GitHub;
- the release changelog is also published on GitHub;

Further details on this activity are provided in the gCube Distribution Wiki at GitHub wiki page.

3.2 ETICS IMPROVEMENTS

ETICS is a tool, developed in past EU research projects[1] and currently operated and maintained by ENG, used for the automation of gCube components build, integration and testing. The tool is used by developers to configure their components releases and by release managers to organize and control the release of the entire gCube system.

During the BlueBRIDGE project, some functionalities have been added to the ETICS tool in order to improve the efficiency and the automation of the gCube releases integration activity. In particular, the following new functionalities are available:

1. **ETICS Issues**: a web-based interactive report of issues of components under release. The list is automatically populated based on the outcomes of builds of candidate components. The issues are related to compilation errors (e.g. missing dependency, deprecated dependencies), gCube quality and distribution guidelines (e.g. missing or incomplete README file). The report has been proved to be very useful during gCube releases to promptly detect problems and solve them before the closure of releases;
2. **ETICS Continuous Building**: the capability of triggering builds of single components when either the source code of the component changes (i.e. commit of changes to the project SVN) or the

[1] http://cordis.europa.eu/project/rcn/86604_en.html

configuration of the component has been modified in ETICS. These builds are useful to promptly verify the integration of the component in the release and detect issues;

3. **ETICS ModelSync**: the capability of ETICS to auto-configure itself for building gCube components based on meta-data found in the component source code. This functionality reduces the effort for developers to manually configure and maintain components in ETICS;

Details on the newly added features and an operative user guide for gCube developers are available at ETICS wiki page.

3.3 JENKINS

At the beginning of BlueBRIDGE project an activity started (ref. #948) concerning the feasibility and the needed effort to switch from ETICS to Jenkins for the integration of gCube software. Jenkins is the main tool adopted in the Java's world for the Continuous Integration and Continuous Delivery. Developers can integrate their work by applying continuous merge for the changes made to the project. Jenkins allows an automatic build every time the developers commit their jobs and in real time to understand if the developers' commit corrupt the software product. The same hardware and software requirements has been used to compare ETICS with Jenkins:

- same virtual machine hosted by ENG (bluebridge);
- same operating system (CentOS);
- same Servlet Container (Tomcat);
- same project (a Jenkins instance has been configured at ENG infrastructure and configured to build some gCube components as preliminary tests to switch to the new integration system).

The version used for Jenkins is 1.6.09, the ETICS version used is 1.8.2-0.
The main feature analysed regarding the management:

- Configuration Project (How to model gCube in Jenkins, if possible);
- Role and Permission (Security Management);
- Multi-branch versioning;
- Artefacts and Downstream/Upstream;
- SCM support;
- Backup Management;
- Installation.

More details about the analysis are hosted at Jenkins wiki page.

The analysis concluded that, although Jenkins is the Software Integration Tools most used in the Java's world community,

- For the version control is not possible to keep track about the version of the jobs.

- Multi-branch version is not supported as well as

- Use of the downstream/upstream jobs it's very simple but the risk is to repeat the build more and more time (in particular for the job with no specific configuration but it recalls other parametrized trigger plug-in).

4 TESTING ACTIVITIES RESULTS

The session provides the results for the testing activities for the releases org.gcube.4.0.0 and org.gcube.4.1.0.
As reported at:

- Functional Master Table for org.gcube.4.0.0 only 91% of the portlets tested, passed the test (on 22 portlets) , minus of the 1% is fails;
- Functional Master Table for org.gcube.4.1.0 all of 33 portlets tested, passed.

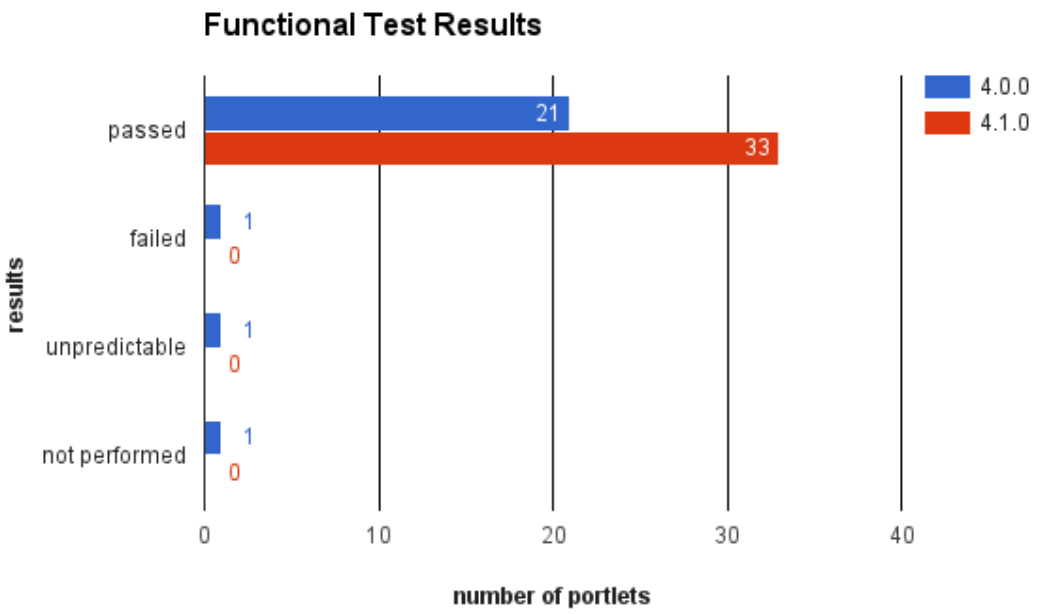

Figure 2. Functional Test Result

5 RELEASE HISTORY

The section provides a summary of the software releases integrated, tested and rolled-out from the first month of the project , September 2015 (more details are available Software Integration and Distribution Release Log wiki page).

The following releases were provided during this period:

- org.gcube.3.9.0 (#581),
- org.gcube.3.10.0 (#1355),
- org.gcube.3.10.1 (#2280),
- org.gcube.3.11.0 (#2071 based on Smart-Gears and #2224 based on gCore),
- org.gcube.4.0.0 releases (#2371 based on Smart-Gears and #4279 based on gCore),
- org.gcube.4.10 (#4316 based on Smart-Gears and #4976 on gCore),
- org.gcube.4.1.1 (#5816).

The following table summarizes the main results for these releases (more details are hosted at Software and Distribution Release Log wiki page):

Table 1. Software Releases

Release	Start	Duration (gg)	Total components	New/Update
3.09.0	28 /09/15	64	574	134 (16 new + 118 upd)
3.10.0	04/12/15	67	536	146 (49 new + 86 upd - 11 drp)
3.10.1	17/02/16	51	525	546(4 new + 86 upd + 11 drp)
3.11.0	04/04/16	51	535	108 (17 new + 84 upd + 7 drp)
4.0.0	20/06/16	37	544	210(29 new + 161 upd + 20 drp)
4.1.0	12/09/16	53	575	231(46 new + 170 upd + 15 drp)
4.1.1	21/11/16	2	575	2 (2 upd)

As represented in the following figure, the release with the highest value in term of total components released is the org.gcube.4.1.0 and org.gcube.4.1.1 version: 575; on the contrary , the release org.gcube.3.10.1 has lowest value of components released: 536.

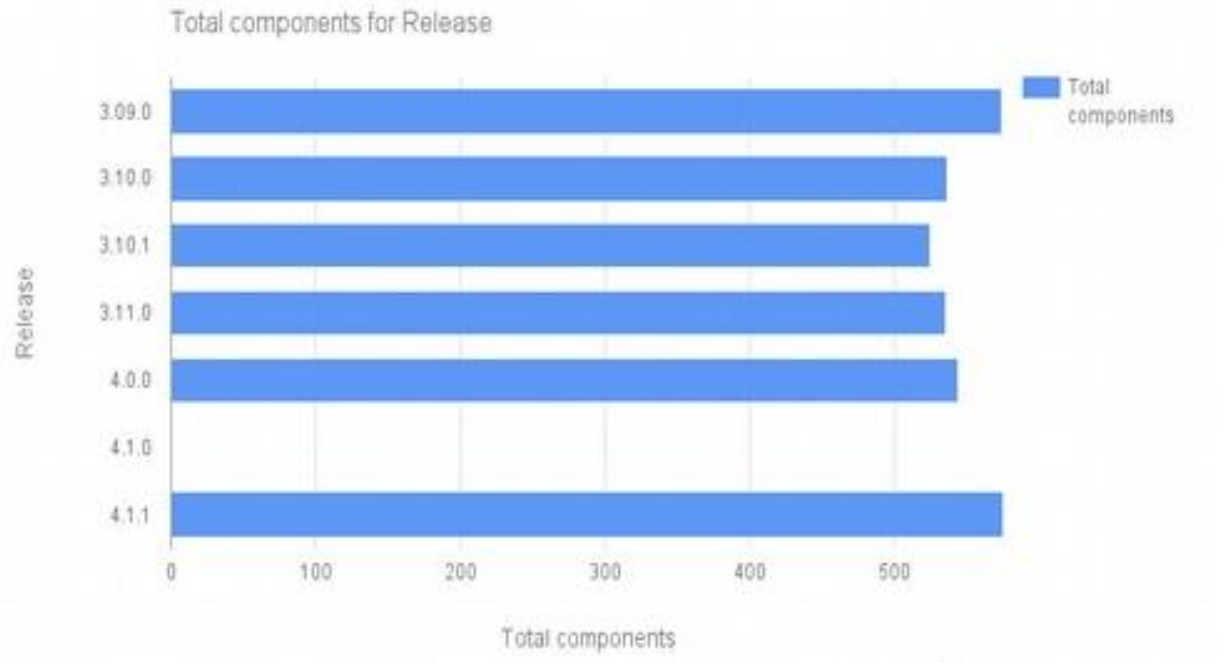

Figure 3. Number of components per release

The following figure shows that the release org.gcube.4.1.0 has a highest value in terms of total components "updated": 170 on 231 released ; on the contrary, org.gcube.3.11.0 has the lowest value in terms of total components "updated" : 84 on 108 released. The release org.gcube.4.0.0 has highest value of components dropped:20 on 161 released; org.gcube3.10.1 and org.gcube.4.1.1 have lowest value of components dropped:0. The org.gcube.3.10.0 release has highest value of "new" component":49 on 146 released; on the contrary org.gcube.4.1.1 (it's a maintenance release) has the lowest value of total components "new": 2 on 2 components released.

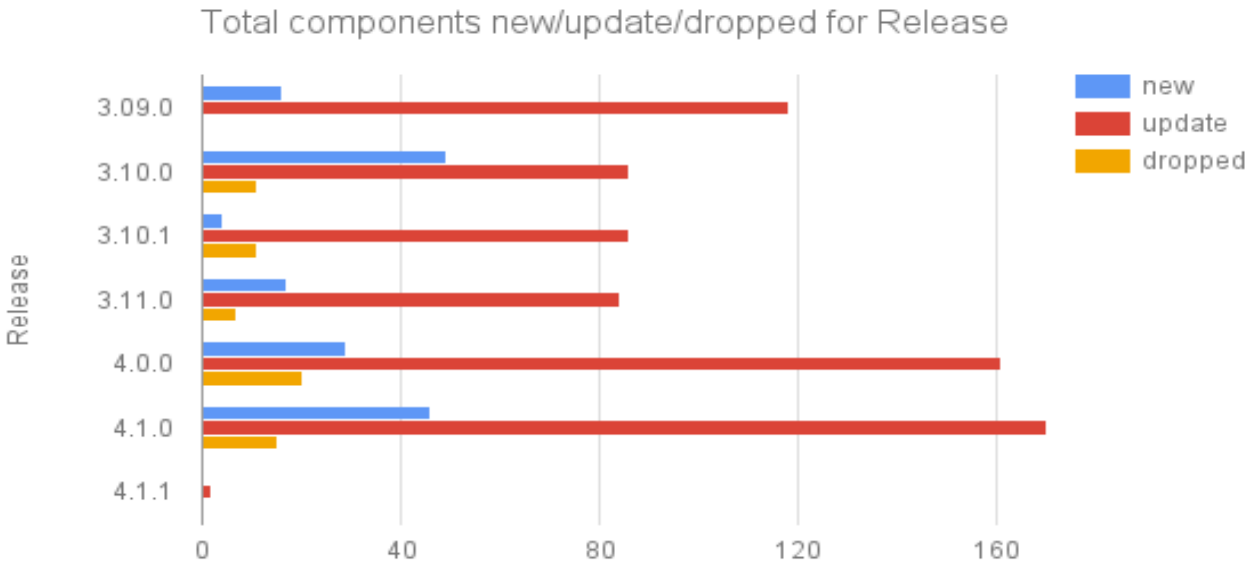

Figure 4. Components per release: new, updated and dropped

The release with the lower value in term of duration of day is the org.gcube.4.1.1: 2 components released in 2nd days; on the contrary, the release org.gcb3.10.0 has higher value : 146 components released in 67th days.

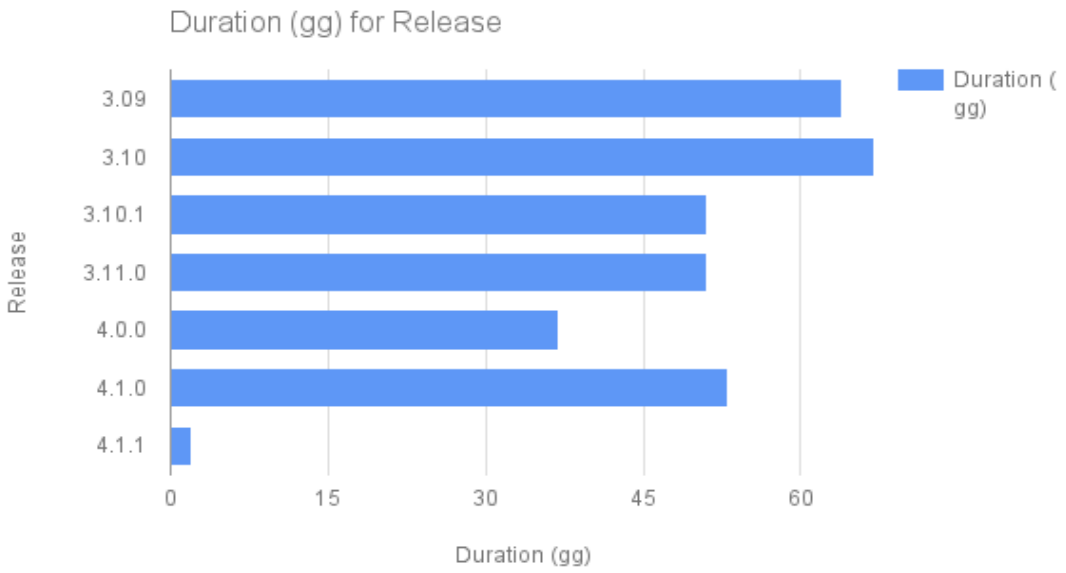

Figure 5. Release Cycles Duration

5.1 GCUBE 3.9.0

The integration activities for gCube 3.9.0 (#581) lasted 64 days. During the release:

- 33 components were removed and 134 updated;
- a list of components that need to be re-factored, because they depend on obsolete components and a new versioning system is used, is provided.

5.2 GCUBE 3.10.0

The integration activities for gCube 3.10.0 (#1355) lasted 67 days. In total, 42 obsolete components were removed and 146 components have been updated.

5.3 GCUBE 3.10.1

A maintenance release (org.gcube.3.10.1, #2280) is provided to fix logging configuration for some components and enforce the update of all the Readme files of the project software components to be compliant with a new format README specification.

It is released in 57 days and 510 components were updated.

5.4 GCUBE 3.11.0

org.gcube.3.11.0 (#2071) represents a turning point in the management, development and implementation of the gCube components. From now it will have two parallel and distinct release: one based on Smart-Gears and one based on gCore. The release includes 234 components updated (in 51 days) and published (released source code) on GitHub .

5.5 GCUBE 4.0.0

org.gcube.4.0.0 (#2371) release has been integrated, tested and deployed during July 2016. This release included a major upgrade of the portal technology moving to Liferay 6.2. All portlets have been migrated to the new version. The release includes 40 new components and 134 updated in 37 days (more details in the table and Sotware Integration and Distribution Release Log.

5.6 GCUBE 4.1.0

org.gcube.4.1.0 (#4316) release is integrated, tested and deployed during October 2016. The details for the main changes are hosted on the gCube 4.1.0 section GitHub page. The release includes 46 new components and 170 updated in 53 days (more details in the table and Sotware Integration and Distribution Release Log.

5.7 GCUBE 4.1.1

A maintenance release (org.gcube.4.1.1, #5816) is performed to fix a critical issue in the newsfeed portlet. As reported in the GitHub page :

"*The problem was causing the News Feed portlet to download large size images contained in posts instead of the small (thumbnail) size. By fixing this bug the load time of the News Feed portlet and the pages containing it has sensibly improved.*"

The integration activities lasted 2 days and 2 components are updated.

5.8 GCUBE HEAD

org.gcube.HEAD provides the development version for the gCube project. A nightly build is provided in order to help the developers to keep up with the rest of the system and project developments: an BRT instance is installed to provide the build results from the web hosted by CNR (as reported in the BTRT wiki page). The gCube is steadily updated with the last development version of the gCube components. Starting to gCube.3.11, the org.gcube.HEAD release is divided in two version:

- org.gcube.HEAD (SmartGears based),
- org.gcube.HEAD-gcore (Gcore based).

During the first period of the project, 316 builds were executed and the developers have been contacted directly to fix any issues met.

REFERENCES

[1] BlueBRIDGE project wiki - https://support.d4science.org/projects/bluebridge/wiki

[2] gCube wiki - https://wiki.gcube-system.org/gcube/About_gCube

[3] ETICS tool website – http://etics.res.eng.it/

[4] gCube website - https://www.gcube-system.org/

[5] BTRT instance at CNR – http://eticsbuild2.research-infrastructures.eu/BuildReport/home/AllBuilds

[6] Release Log at Work Package 4 - https://wiki.gcube-system.org/gcube/Software_Integration_and_Distribution:_Release_Log

[7] Developer's Guide - https://wiki.gcube-system.org/gcube/Developer's_Guide

[8] Administrator's Guide - https://wiki.gcube-system.org/gcube/Administrator's_Guide

[9] User's Guide - https://wiki.gcube-system.org/gcube/User's_Guide

[10] The BlueBRIDGE Issue Tracker - https://support.d4science.org/

[11] European Union Public Licence V. 1.1 (EUPL © the European Community 2007) - https://joinup.ec.europa.eu/sites/default/files/ckeditor_files/files/eupl%20v%20%201%201%20-%20EN.txt.txt

[12] Jenkins Official Site https://jenkins.io/

[13] The Administering Jenkins documentation

https://wiki.jenkins-ci.org/display/JENKINS/Administering+Jenkins