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: Final Report***
Deliverable No.	***D4.4***
Delivery Date	***January 2018***
Authors	***Maria Antonietta di Girolamo (ENG)*** ***Gabriele Giammatteo (ENG)***

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.	D4.4
Deliverable Title	Software Release Activity: Final Report
Contractual Delivery Date	January 2018
Actual Delivery Date	March 2018
Author(s)	Gabriele Giammatteo (ENG), Maria Antonietta Di Girolamo (ENG)
Editor(s)	Gabriele Giammatteo (ENG), Maria Antonietta Di Girolamo (ENG)
Reviewer(s)	Pasquale Pagano (CNR)
Contributor(s)	Daniele Pavia (ENG)
Work Package No.	WP4
Work Package Title	VREs Deployment and Operation
Work Package Leader	ENG
Work Package Participants	CITE, CNR, ENG, UOA
Distribution	Public
Nature	Other
Version / Revision	1.0
Draft / Final	Final
Total No. Pages (including cover)	26
Keywords	software integration, packages, distribution, release, testing

DISCLAMER

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/).

GLOSSARY

ABBREVIATION	DEFINITION
Apps	Applications
BlueBRIDGE	Building Research environments for fostering Innovation, Decision making, Governance and Education to support Blue growth
BTRT	Build and Test Report Tool
Comps	Components
CRT	Component Release Ticket
Dev	Developer
ECC	ETICS Component Configuration
EPC	ETICS Project Configuration
ESC	ETICS Subsystem Configuration
ETICS	e-Infrastructure for Testing, Integration and Configuration of Software
PRT	Project Release Ticket
RMAN	Release Manager
SMan	Subsystem Manager Role
SRT	Subsystem Release Ticket
SubSys	Subsystems
SW	Software
TTEAM	Tester team
VRE	Virtual Research Environment

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. It focuses on the second reporting period (from M16 to M29), while the same report for the first reporting period is contained in deliverable D4.2 “Software Release Activity: Interim Report” delivered at M14. 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 for each release. Furthermore, it contains some statistics about the release activities throughout the entire project.

EXECUTIVE SUMMARY

This document reports the activities related to the releasing and testing of the software produced and maintained during the second reporting period of the BlueBRIDGE project (task T4.4).

In order to improve and maintain the efficiency of the procedures and tools used in the release of the software produced in the project, a set of activities have been undertaken in this period. The most relevant ones described in this report are a) the update of the release and testing procedures, b) the introduction of new deployment tests, c) the migration to Java 8 and the update of Maven tools, d) the planning of the migration of the source code to Git and e) the improvement of the software release management tools interface.

During this reporting period, 12 new releases of the software have been successfully integrated, tested and distributed. In total, 894 new or enhanced versions of components have been integrated and released in this reporting period. On average, the duration of the release is 40 days with some variations (less from the introduction of the new release procedure) due to external time constraints (e.g. holiday breaks).

1 INTRODUCTION

This document reports the activities related to the releasing and testing of the software produced and maintained during the second reporting period in the BlueBRIDGE project (task T4.4).

Since the development effort in BlueBRIDGE – distributed in Blue Commons (WP9 and WP10) and all the Blue Pillars (WP5 to WP8) - flows into the gCube System[1] codebase, the term "gCube release" will be used to refer to the release of the software produced in BlueBRIDGE.

The activities reported can be categorized in two different types: a) activities aimed to integrate new gCube releases and b) activities aimed to maintain and improve the procedures and tools used during the integration of gCube releases.

The importance of having efficient procedures and tools for the release activities is crucial in the project given a) the big number and the high complexity of the components implemented, b) the high number of developers and organizations involved, and c) the tight scheduling for the distribution of new functionalities. To this end, during the second reporting period, it is worth to report the following major activities:

- the update of the release procedure to parallelize as much as possible the integration, testing, and distribution activities, described in section **Error! Reference source not found.**;
- the update of the functional testing procedure to improve the coordination between tester team and developers, described in section **Error! Reference source not found.**;
- the introduction of a new type of automated deployment testing based on Ansible[2], described in section **Error! Reference source not found.**;
- the migration from Java 7 to Java 8, described in section **Error! Reference source not found.**, and the update of Maven building tools, described in section **Error! Reference source not found.**;
- the improvements of the release management tools described in section **Error! Reference source not found.**;
- the design of a migration solution of the gCube codebase to Git, described in section **Error! Reference source not found.**.

These improvements allowed to achieve considerable improvements in terms of number of releases rolled-out and tests carried out during the reporting period when compared with the previous reporting period. A summary of the gCube releases carried out is available in section **Error! Reference source not found.**.

The same report for the first reporting period is available in the deliverable D4.2 "Software Release Activity: Interim Report" delivered at M14.

[1] https://www.gcube-system.org/

[2] https://www.ansible.com/

Since this is the final report on task T4.4 activities, two final sections with statistics and trends about the entire project are available in section **Error! Reference source not found.** and 6.3**Error! Reference source not found.**.

2 RELEASE PROCEDURE UPDATES

During the second reporting period, the release procedure has been heavily revised in order to improve the efficiency of the release cycle. The previous release procedure was inherited from the previous iMarine project with few adjustments. During the project, the new requirements in terms of gCube releases outcomes and scheduling timeframes introduced delays in the release scheduling and several issues at integration and testing time. This had to be fixed to avoid delays in the development activities.

For this reasons, a refactoring of the release procedure has been undertaken aiming at improving the efficiency of the gCube release activities. The main pillars for the new procedure are:

- release cycle duration fixed at six weeks with no pauses between releases;
- creation of three categories of components (i.e. enabling, common apps, and community apps) with differentiated deadlines for each group;
- the integration, update of the preproduction, testing, and update of the production activities start independently for each group.

In particular, the categorization of components in groups and its management during the release contributed (in most of the cases) to avoid delays caused by issues to few components that block the entire roll-out of the release.

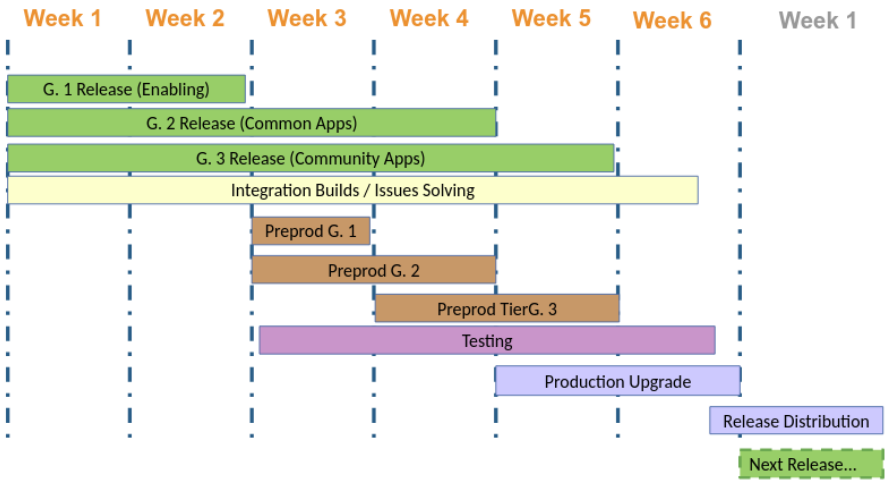

Figure 1: Release Cycle Gantt chart

Figure 1 depicts the scheduling of the different release phases for each group of components. A more in detail description of the procedure and the operative instructions for developers and release managers are maintained in the gCube Distribution wiki at:

https://wiki.gcube-system.org/gcube/Major/Minor_Release_Cycle_procedure.

3 INTEGRATION TOOLS UPDATES

This section describes the activities related to the update of the tools used during the integration phase of the gCube releases. In particular following updates will be reported:

- improvements to the ETICS GUI (section **Error! Reference source not found.**);
- upgrade of the gCube Java version (section **Error! Reference source not found.**);
- update of the gCube Maven building system (section **Error! Reference source not found.**);
- update of the source code repository technology (section **Error! Reference source not found.**).

3.1 ETICS GUI IMPROVEMENTS

A new option has been added to ETICS to allow users to hide a module from the GUI (see Figure 2). It has been used to hide from the ETICS GUI all the gCube modules that have been dismissed or replaced by new modules. In fact, over the years, given the high development rate in gCube, the number of dismissed modules increased and searching or browsing through the modules in the ETICS GUI became difficult. Hiding all the dismissed modules, improved the usability of the GUI.

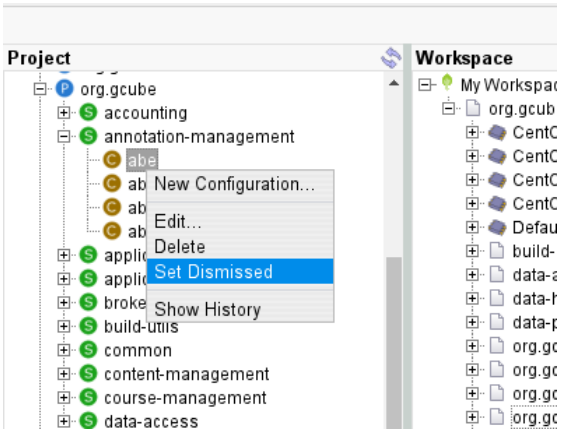

Figure 2: Set Dismissed modules option in ETICS

More details and usage instructions for this functionality are available on-line at:

https://wiki.gcube-system.org/gcube/ETICS#Dismissed_Modules.

3.2 JAVA 8 MIGRATION

An important technological update has been achieved during this reporting period: the migration of the D4Science infrastructure and most of gCube codebase to Java 8. The necessity for this update was justified by two main reasons:

- Java 7 support stopped on April 2015 and updates and security fixes were not distributed any more.
- Java 8 introduced several new functionalities in the language (e.g. Lambda functions, Stream API) that couldn't be exploited without migrating to the new Java version.

The migration to Java 8 has been achieved following different steps: a) the update of the infrastructure nodes with the Java 8 runtime, b) the build of Java 8 artefacts, c) the testing (integration, deployment and

functional) of the new artefacts with the Java 8 runtime and finally d) the deployment of the artefacts in the infrastructure. This has been done for all the three D4Science infrastructures - development, preproduction, and production - in sequence avoiding in this way any malfunction of the production infrastructure.

The entire migration required a considerable effort and coordination of all release teams: developers, release managers, testers, and infrastructure managers.

During the migration, a dedicated wiki page on the gCube Distribution wiki has been created and updated regularly to build and share a knowledge base on the migration from Java 7 to Java 8. The page documents all the major integration and deployment issues encountered and the solutions found. It can be consulted at:

https://wiki.gcube-system.org/gcube/GCube_Java_8_migration_guide.

The major issue during this activity was related to an incompatibility of a core third-party library (i.e. *xerces*) used in the old gCube services framework (i.e. gCore) with the new Java 8 runtime. Since the effort to update the affected source code was judged too high, the solution was to keep the old gCore services (very limited in number and deprecated till the planned decommission planned for June 2018) building and running with Java 7.

3.3 MAVEN UPDATES

During the second year of project, the Maven templates and the Maven parent project (the basis for all gCube components) have been updated to simplify their usage and adapt to the new project needs.

The first important change is the introduction of a set of properties to specify the Java version to use to build the components. This update was necessary to support the switch to Java 8 (section 3.2). In fact, given the fact that gCube is currently split in two releases (the gCore-based release and the SmartGears-based release) that build with two different Java versions (Java 7 and Java 8 respectively), a method to make the Java version parametric during the build was necessary. The properties are defined in the maven-parent and defaults to Java 8. This system is flexible and it allows the overriding of the Java version both in the single components pom.xml file or in the command line during the build. Further information can be found in the gCube Distribution Wiki at:

https://wiki.gcube-system.org/gcube/Developing_gCube_Maven_Components#Java_Version.

The second big update was the rationalization of the methods to create the Maven artefacts during the build of the components. In fact, during the years some components started to add in their pom.xml the configuration to build new artefact types (e.g., jar-with-dependencies, uberjar) for new identified needs at deployment time. Since this requirement was common to multiple components, it has been decided to move the configuration to generate these additional artefact types in the maven-parent, to make it available to all components, and to standardize the way they are created. The artefact types standardized in this way are:

- uberjar: a jar that contains the classes of the components plus all the classes of the dependencies. It is not enabled by default and it requires an explicit invocation of the maven-assembly-plugin in the component pom.xml to activate it;

- source-package: a package that contains the source code of the component plus some metadata files (e.g. README, LICENSE). Generation of this artefact is activated by the property "generateDistribution";
- servicearchive: an alternative way of creating the servicearchive that uses centralized README and LICENSE files. Can be activated by invoking the maven-assembly-plugin in the component pom.xml;
- javadoc: generates the documentation of the component Java classes. Enabled by default for all components.

Further details can be found in the gCube Distribution Wiki at:

https://wiki.gcube-system.org/gcube/Developing_gCube_Maven_Components#Artifact_Types.

3.4 GIT MIGRATION

Since the beginning of the project an activity to migrate the gCube codebase from the current SVN-based source code repository to a Git-based repository has been started. This choice is justified by several reasons, including: a) the increasing size of the gCube code base makes it more efficient if managed with Git, b) Git is newer, faster and it offers more collaboration functionalities, c) seamless integration with GitHub (where the gCube source code is already published).

The migration process has to take into account the need to preserve the history of the codebase (i.e. all SVN commits has to be migrated, not only the latest version) since the gCube codebase is an open source and widely used framework. This is possible using some existing importing tools (e.g. git-svn, subgit), but the tests revealed that this operation is very slow and therefore, given the size of the gCube codebase and its update rate, it was not possible to think to a one shot migration of the entire code because it would stop the development activity for too much time.

This constraint led to the design of a solution to migrate single gCube components autonomously. This approach allows to integrate and distribute gCube releases also with some components already migrated and some not. It is a more complex solution because it must guarantee the coexistence and the support for both types of repositories, but allow to split the effort of the migration over multiple developers and over a longer period of time.

The complexity of this solution and the effort required led us to complete the migration within the scope of BlueBRIDGE project only partially. This does not represent an issue since the gCube consortium agreed to continue the activities without any interruption after the end of the BlueBRIDGE project itself. Nevertheless, a document reporting in detail all the work done on this activity including the design of the technical solution, changes to integration tools identified and a migration strategy is available on the gCube Distribution Wiki at:

https://wiki.gcube-system.org/gcube/GCube_Git_Migration.

4 RELEASE TESTING UPDATES

This chapter describes the new testing procedure and report on the improvements of the testing activities after the changes of the new release cycle procedures.

4.1 FUNCTIONAL TESTING PROCEDURES

The functional testing procedure received an important improvement during this reporting period. In fact, the old tool used to manage and coordinate the tests (a shared wiki page) has been replaced by a dedicated tracker on the BlueBRIDGE issue-tracking tool, powered by Redmine.

This improvement allowed to track in a better way the status of each functional test with additional information such as the responsible tester and developer, the deadlines, the affected components, the priority, etc. Since Redmine is also used for tracking all the gCube development activities and issues, the tickets for the tests can easily be linked with additional related tickets (e.g. integration issue, release tickets).

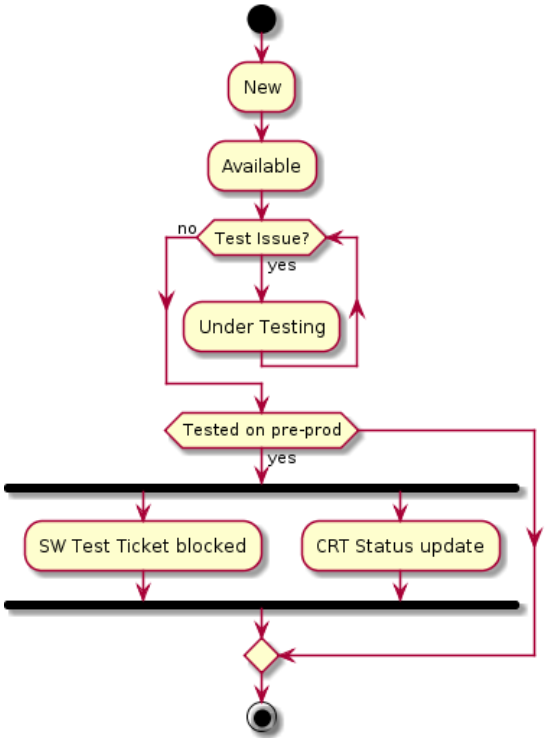

Figure 3: Functional testing ticket statuses

Figure 3 depicts the workflow of the possible statuses of a test ticket.

The testing procedure starts with the creation of the *Master Ticket* and (as children tickets) one ticket for each expected test. This is done by the Test Manager that also assigns each test ticket to a tester. Then, after the software under test has been deployed in the preproduction infrastructure, the responsible tester executes the test reporting the results on the ticket.

The functional testing procedure and the tools used are described in detail in the gCube Distribution wiki at:

https://gcube.wiki.gcube-system.org/gcube/Functional_Testing.

4.2 ANSIBLE DEPLOYMENT TESTING

Starting with gCube release 4.5.0, an automatic deployment test of a basic gCube infrastructure has been implemented. It is executed immediately after a daily integration build: the built artefacts of the enabling services of the gCube framework are automatically deployed and the results reported in the daily builds reporting tool (i.e. BTRT).

At the core of this solution lies Ansible. A set of Ansible roles have been written, each specialized in deploying a certain gCube artefact: gcube-authorization, gcube-collector, gcube-portal, gcube-resourcebroker, gcube-softwaregateway, gcube-base (gcube distribution), gcube-notifier, gcube-registry and gcube-resourcemanager.

A simple python web-server, called Orchestrator, listens for incoming notification of the new daily build. When a new notification is received, the orchestrator downloads the list of built artefacts, swipes the testing infrastructure and invokes Ansible. Each Ansible role (listed above) deploys the artefact on a freshly installed gCube container, waits for the gCube container to start and makes a request to the service to check that it has been correctly deployed. When all the roles have been executed, the results are fetched from the daily build reporting tool (i.e. BTRT) and shown in the “Deploy Test” column (Figure 4).

Figure 4: BTRT deployment test result

5 RELEASE DISTRIBUTION UPDATES

During the second reporting period, the activity of updating the gCube distribution - started with the publication of releases on GitHub (see D4.2, section 3.1) – continued with the aim to rationalize all the scripts and services developed over the years and consolidate them in a new service, the Software Publisher.

The architecture of the new service is based on the creation of a new distribution file (in xml) that contains, for each package, the full set of information about that package including links to source code and packages, authors, changelog, readme, etc. This file is created at the end of each nightly build by collecting information from the built packages and from the ETICS model data. Then a chain of processors is able to load the distribution file and perform different transformations (e.g. generation of other partial distribution files, publication on an external repository).

The new service replaced most of the custom distribution scripts and simplified their creation. The processors implemented so far perform the following actions:

- generation of the Ansible variable file (see section **Error! Reference source not found.**) needed to run deployment tests;
- generation of distribution file for Zenodo. It is the input to the Zenodo Publisher (see section **Error! Reference source not found.**);
- generation of the legacy distribution file, still used to publish the gCube releases on the gCube website;
- generation of the release notes file;
- preparation of the GitHub publication (see D4.2, section 3.1);
- generation of list of locked configurations, used by BTRT to display the build results.

These processors currently are executed as part of the nightly build process, but the Software Publisher can also be exposed as a gCube empowered SmartGears service to enable the possibility to explicitly invoke the processors remotely.

5.1 ZENODO PUBLICATION

In the context of the distribution of gCube software, an important activity was the publication of gCube software in the Zenodo portal.

Zenodo is a portal (launched in May 2013) that collects outputs from researches in all fields of science to promote open access and open data. It is supported by the OpenAIRE initiative and developed and hosted by CERN. Each outcome uploaded (called Deposition in the Zenodo terminology) is stored by Zenodo along with a rich set of metadata. Zenodo assigns to each deposition a unique Digital Object Identifier (DOI) to make the upload easily and uniquely citable.

Given the number of gCube components to publish at every gCube release and the number of information to publish for each component, it has been decided to fully automate the publication process exploiting the REST API offered by Zenodo. The implemented solution creates new Zenodo deposition for each component for each gCube release and uploads the corresponding source code. It is able not only to add,

but also edit already existing depositions (in case any information changes). At the time of writing, more than 3700 gCube depositions have been published on Zenodo.

Further information, including the full list of published information for each component, is available in the gCube Distribution wiki at:

https://wiki.gcube-system.org/gcube/Zenodo_Publication.

6 GCUBE RELEASES

This section provides a short changelog and description of activities related to each gCube release rolled-out during the second reporting period. A summary of the releases of the first reporting period is available in deliverable D4.2 “Software Release Activity: Interim Report”.

A more detailed summary is available in the gCube Distribution wiki at:

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

In addition, section 6.2 and 6.3 reports some statistics related to the gCube release and testing activity during the entire BlueBRIDGE project.

6.1 SECOND REPORTING PERIOD RELEASE HISTORY

6.1.1 GCUBE 4.2.0

The main changes in the gCube release 4.2.0 involved the following components: Accounting Manager Interface, SmartGears, Resource Registry Publisher, Authorization 2.0, data-transfer, data-miner and data-analysis. Various defects have been solved too.

In total 121 components were been updated/added and the integration of the release took 19 days.

6.1.2 GCUBE 4.2.1

This maintenance release fixed issues involving the following components for the: DataAccess, DataCatologue, Portlet User and Widgets, Information System and Portal. The maintenance release took 8 days in total.

6.1.3 GCUBE 4.3.0

The main changes in the gCube release 4.3.0 involved all gCube subsystems (several improvements and bug fixing). In total 12 new and 189 updated components have been released. The release took 59 days in total. The very long duration of this release highlighted the limits of the old release procedure and was the main input for its refactoring (see section **Error! Reference source not found.**).

The new functional testing procedure has been introduced in this release as described in the chapter “Functional Testing procedures” (section 4.1).

6.1.4 GCUBE 4.4.0

This release, the first rolled-out with the new release procedure in place, took in total 40 days. It contains several improvements and bug fixing for several components including Smartgears, AuthZ Framework, Index and Search, Geoanalytics.

During this release, an analysis of all components has been carried-out with the aim of removing from the release obsolete, deprecated and unused components. In total 100 components have been removed with an improvements of the building times of about two hours.

6.1.5 GCUBE 4.5.0

The gCube release 4.5.0 contained several improvements and bug fixing for most of gCube subsystems. The work for cleaning-up the release from unused components continued. In total 10 new and 94 updated components have been released and 16 components have been removed. The release lasted for 40 days.

6.1.6 GCUBE 4.6.0

Starting from this release, the integration builds of gCube started to generate artefacts in Java 8 (see section **Error! Reference source not found.**). This required a change and, therefore, the release of few additional components in order to solve incompatibility issues.

During the integration activity of this release a major issue occurred that forced to revise the release plan. The ETICS service (used to execute the integration builds) has been unavailable for a long period of time. This forced the release managers to set-up an alternative process to execute integration builds “manually” (one by one the single components). This alternative worked fine, but due to the high effort required, has been decided to release only the community apps components.

In total, the release counted 13 new, 63 updated and 3 deleted components and took 43 in total.

6.1.7 GCUBE 4.6.1

This maintenance release has been created specifically to release all components that couldn’t be released during gCube 4.6.0 due to the issue with ETICS, mostly enabling and common apps components. In total the release shipped 14 new and 69 updated components. The release took 23 days in total.

6.1.8 GCUBE 4.7.0

The main changes in the gCube release 4.7.0 involved the following subsystem: DataAnalysis, User Portlets, Information System, GeoExplorer. The release integration lasted 40 days.

6.1.9 GCUBE 4.7.1

This maintenance release fixed a critical bug found in the production portal that causes in some circumstances users to be deleted from a group when they are assigned to new groups and fixed the group email template formatting. The duration of this release was of 3 days.

6.1.10 GCUBE 4.8.0

The main changes in gCube 4.8.0 involved the following subsystems: Smart Executor, Data Analysis, User Portlets, Portlet Widgets, Data Transfer, Common and Information System. In total 4 new and 47 updated components were integrated in the release that lasted 49 days.

Starting from this release, a codename (inspired by the name of an ocean or see) has been associated to each release. The first one is *Atlantic.*

6.1.11 GCUBE 4.9.0

gCube 4.9.0 (codename *Baltic*) was planned to be shorter than normal because of the Christmas break. For this reason, only Community Apps or high priority components were accepted.

The release lasted only 22 days in total with 3 new and 35 updated components.

6.1.12 GCUBE 4.10.0

gCube 4.10.0 (codename *Caribbean*) has not been completed yet at the time of writing, therefore no final statistics are available. So far, 12 new and 49 updated components on 15 different subsystems have been released. It should be concluded by the end of M30.

6.1.13 GCUBE HEAD

During the periodic report regular nightly builds (for a total of more than 350) of the development version of the gCube software were executed to solve problematic situation. Developers were contacted directly or by opening issue tickets asking for fixing their components.

6.2 GCUBE RELEASES STATISTICS

During the BlueBRIDGE project 19 gCube releases have been rolled-out. A summary of all the releases with duration and number of changed subsystems and components is reported in Table 1: gCube releases in BlueBRIDGE. From this data, three charts have been extracted (Figure 5, Figure 6 and Figure 7) to help the analysis of the data.

RELEASE	START	DURATION (days)	TOT. SUBSYSTEMS	CHANGED SUBSYSTEMS	TOT. COMPONENTS	CHANGED COMPONENTS
3.9	28 September 2015	64	28	17 upd	574	16 new, 118 upd
3.10	04 December 2015	67	27	22 upd	536	49 new, 86 upd, 11 del
3.10.1	17 February 2016	51	26	26 upd	510	4 new, 487 upd, 19 del
3.11	04 April 2016	51	24	24 upd	535	17 new, 84 upd, 7 del
4.0	20 June 2016	37	21	17 upd, 1 new	544	29 new, 161 upd, 20 del
4.1	12 September 2016	53	26	22 upd, 2 new	575	170 upd, 46 new, 15 del
4.1.1	21 November 2016	53	26	2 upd	575	2 upd
4.2	25 November 2016	19	30	11 upd	585	11 new, 109 upd, 1 del
4.2.1	2 February 2017	8	26	7 upd	586	1 new, 25 upd
4.3	20 January 2017	59	26	17 upd, 1 del, 1 new	582	12 new, 189 upd, 16 del
4.4	20 March 2017	40	26	26 upd	491	9 new, 95 upd, 100 del
4.5	02 May 2017	40	22	22 upd	485	10 new, 94 upd, 16 del
4.6	12 June 2017	43	30	17 upd	495	13 new, 63 upd, 3 del
4.6.1	25 July 2017	23	30	20 upd	502	14 new, 69 upd
4.7	28 August 2017	40	30	16 upd	497	1 new, 58 upd, 6 del
4.7.1	16 October 2017	3	30	1 upd	497	1 upd
4.8	11 October 2017	49	30	18 upd	501	4 new, 47 upd
4.9	28 November 2017	22	30	11 upd	504	3 new, 35 upd
4.10	18 December 2018	57	30	15 upd	516	12 new, 49 upd

Table 1: gCube releases in BlueBRIDGE

The chart in Figure 5 depicts the changed (new, updated or deleted) components for each gCube release excluding the maintenance releases that, by their nature, have values very far from the average of a normal release. As expected, it shows a peak of new and updated components between gCube 4.0.0 and gCube 4.3.0 that corresponds to the most active period (from Jun 2016 to January 2017) for the development activities in BlueBRIDGE. In that period in fact, most of the VREs of the project have been implemented and released to the community. After that period, number of changes tends to decrease due the fact that the project entered in a phase of exploitation of the VREs. It is also worth noting how the total number of components slightly increases in all releases excepting for a sudden fall between gCube 4.3.0 and gCube 4.4.0. In fact, in the latter release a systematic clean-up of the release from abandoned and unused components has been executed (see section **Error! Reference source not found.**).

Figure 6 shows a cumulative chart of the new and updated components during the project. In total, for the BlueBRIDGE project about 1500 new or updated version of components have been released. Of them, 894 components belong to the second reporting period (from gCube 4.2.0).

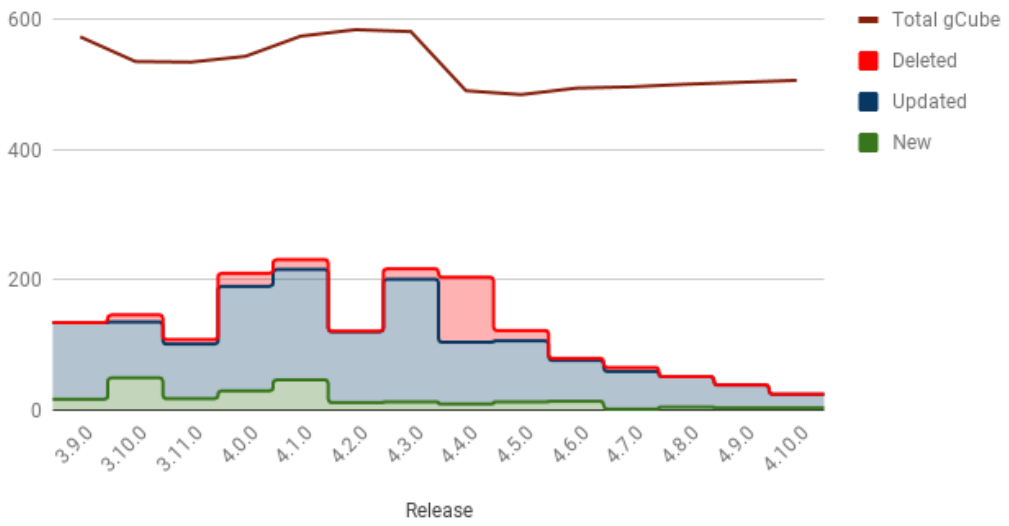

Figure 5: Changed components per release chart

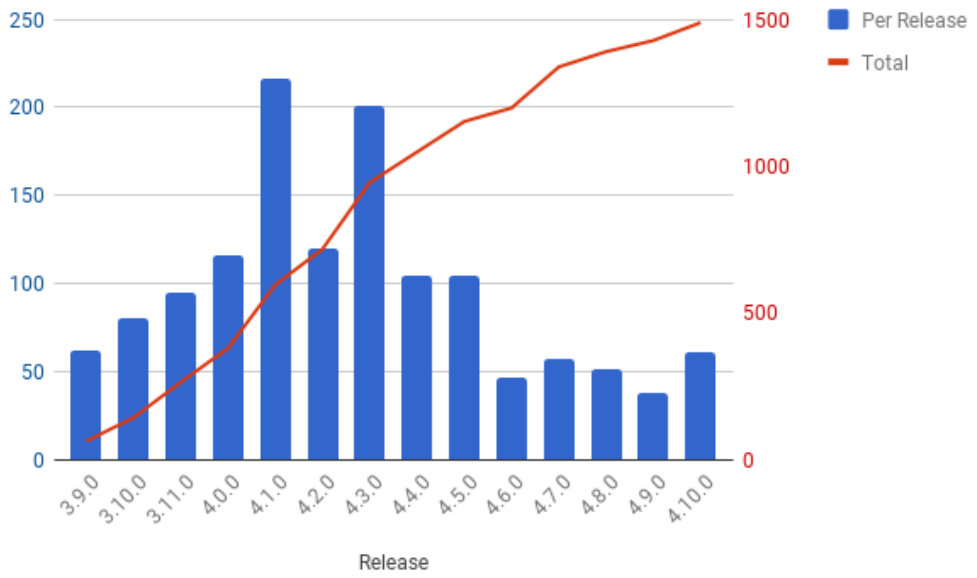

Figure 6: Cumulative number of new and updated gCube components

The chart in Figure 7 shows the duration of each release from the start of the integration builds to the roll-out in production. Maintenance releases are excluded since they follow a very specific scheduling (imposed by the urgency of releasing fixed software) that differs from the planned releases scheduling. The most interesting interpretation of this data is that starting from the introduction of the new release and testing procedures (section **Error! Reference source not found.** and **Error! Reference source not found.**), the duration of the releases became much more stable (close to the 42 days planned in the procedure). The gCube 4.9.0 exception is explained by the Christmas break constraint for this release.

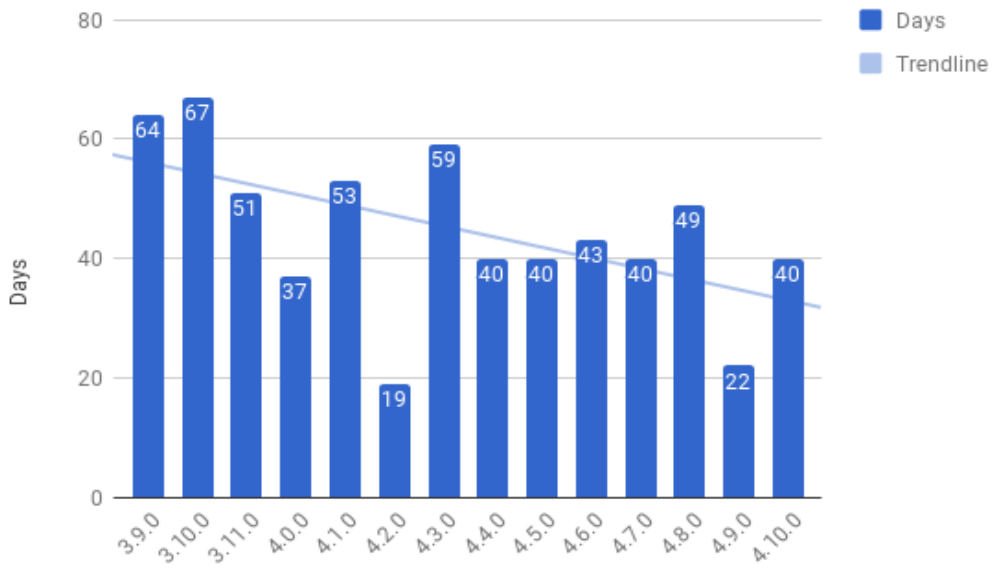

Figure 7: Releases duration chart

6.3 GCUBE TESTING STATISTICS

During the project, the functional testing activities became more and more important to deliver quality software. It became also more challenging to perform in the assigned timeframe since several new and more complex components have been added to release and this increased the interdependency between components (and, as consequence, the probability of errors). For this reason, while in the first reporting period, formal test-plans, shared with all the release teams, have been introduced for each component to test and, in the second reporting period, the functional testing procedure has been revised and improved (see section **Error! Reference source not found.**).

Figure 8 shows the number of new, updated and deleted test-plans for each release. A new test-plan is added every time a new component to test is added to the release and updated every time any functionality of the component changes or a defect is resolved (i.e. regression test). As expected, the update of test-plans was more intense during the first part of the project because of the more intense development activity on gCube components.

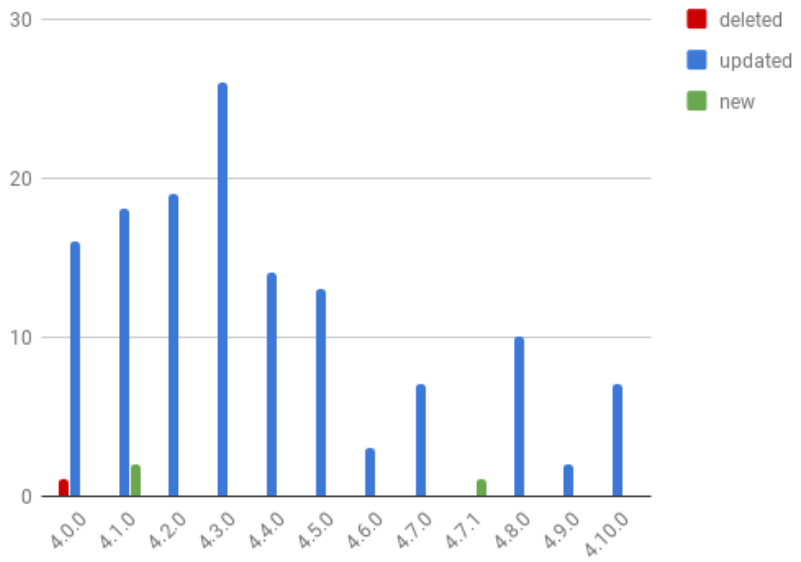

Figure 8: New, Updated and Deleted testplans per release

Figure 9 shows the number of functional test executed and the number of issues discovered during their execution for each release. It must be considered that tests were executed only if the corresponding components were new in the release or changed (a new version released) from the previous release.

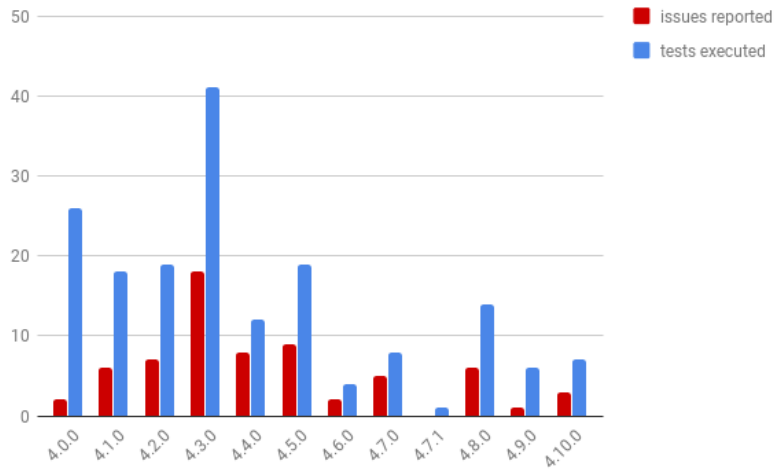

Figure 9: Functional test executed per release

7 CONCLUSION

This document reported the activities performed in the BlueBRIDGE project task T4.4 related to the release of the gCube software during the second reporting period. The same report for the first reporting period is available in the deliverable D4.2 “Software Release Activity: Interim Report” delivered at M14.

The two main procedures used in the task to manage the release and testing activities have been heavily revised in order to improve their efficiency, reduce the duration of the releases, and reduce the number of integration issues. This update, introduced from gCube 4.4.0, allowed to have a more predictable release scheduling with a duration of about 40 days for each release.

The Java version used to build and run all the gCube components has been updated from Java 7 to java 8. This implied first the upgrade of the runtime software in the entire D4Sciecne infrastructure and then the upgrade of the release tools in order to build and deploy Java 8 compatible artefacts.

A technical solution and a migration strategy have been identified for the migration of the gCube codebase from SVN to Git and have been documented.

During this reporting period, 12 new releases of the software have been successfully integrated, tested and distributed. In total, 894 new or enhanced versions of components have been integrated and released in this reporting period. On average, the duration of the release is 40 days with some variations (less from the introduction of the new release procedure) due to external time constraints (e.g. holiday breaks).

REFERENCES

Giammatteo, G. and Di GIrolamo, M.A. (2016) Software Release Activity: Interim Report. BlueBRIDGE Deliverable D4.2