MORPHEMIC

Test report for prototype release

Modelling and Orchestrating heterogeneous Resources and Polymorphic applications for Holistic Execution and adaptation of Models In the Cloud

H2020-ICT-2018-2020
Leadership in Enabling and Industrial Technologies: Information and Communication Technologies

Grant Agreement Number
871643

Duration
1 January 2020 –
30 June 2023

www.morphemic.cloud

Deliverable reference
D4.6

Date
25 November 2022

Responsible partner
7bulls.com

Editor(s)
Katarzyna Materka

Reviewers
Ali Jawad Fahs, Alexandros Raikos

Distribution
Confidential

Availability
morphemic.cloud

Executive summary

This document presents the process of testing MORPHEMIC Release 2.5. Software testing is aimed at evaluating the quality of a program and improving it by identifying any defects and potential problems further resolved by developers. MORPHEMIC is a complex multi-cloud solution, based on Melodic which is a framework that supports automated deployment of both data and the applications processing the data, based on the constraints set by the organisation owning the data and the application. MORPHEMIC contains a variety of modules, created by the cooperation of many people and organizations. The Quality Assurance approach was based on ISTQB standards and the whole consortium was included in the testing part. Therefore, it is crucial to put extra attention to testing, preparing and executing test cases, managing the lifetime of test cases, and reporting issues.

Coordinated test results and resolution of detected problems are reported in a specialised tool. Based on that, this Deliverable contains a report that was created to show the possibilities of fixing issues encountered during the process of moving from Release 2.0 to Release 2.5. Both tests based on previously created test cases and reported issues with resolutions are summarised in this document.

Author(s)
Jakub Pacześ, Joanna Chmielewska

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871643

Revisions

Date	Version	Partner	Description
15.11.2022	0.1 (draft)	7bulls	First Draft
22.11.2022	0.2	7bulls	Second draft
25.11.2022	0.3	7bulls	Final
05.12.2022	1.1	Activeeon	First review
12.12.2022	1.2	7bulls	Adjustments

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 The scope

Testing is an essential part of every huge project created by the cooperation of many groups and people. To ensure quality and reliability, it is crucial to follow all procedures of testing, prepare and execute test cases, report any issues and take care of the whole lifetime of the test case or issue. Test cases created for release 2.5 are both for checking new functionalities and providing regression tests.

All new functionalities introduced during Release 2.5 such as:

- using multiple BYON nodes
- support of the new architecture (armv7 and armv8) of the EDGE nodes
- deployment of resources in OpenStack

needed to be formally tested. BYON functionality was already available and it was extended to more than one on-premises node

The main idea and a core of MORPHEMIC project is a multi-cloud concept and consequently it is extremely important to verify the continuous employability of all the supported cloud providers. In earlier releases, due to availability, popularity and partners interest, most of the testing focused on AWS resources so in this release OpenStack resources were crucial to check project goals. Regression was an important case as there were many fixes for issues reported within the forecasting feature.

Release 2.5 was initially planned for 31th of August 2022, but we encountered problems with OpenStack platform configuration, available resources and networking problems, not allowing us to deploy our test applications on OpenStack resources. Some essential fixes for issues from previous release were crucial, and some of the resources were focused on them as a priority before completing tasks for release 2.5. Due to encountered unexpected situation with environments, the consortium decided to change a scope of release requirements to resolve further complication at time when they appeared to keep the quality of the platform and reduce further workload that may appear when old problems are left behind for further releases. To avoid additional work in the future all issues were addressed and the effective release date was moved to 15th of November 2022.

First part of this document is focused on presentation of MORPHEMIC basic technical information, such as technical requirements of the environment for proper installation of the platform and approach to meet the requirements that were made during tests of the platform. This document also presents information about the documentation part of the test for the MORPHEMIC project.

1.2 Audience

This deliverable is intended for those involved in the software quality assurance process and their outcome:

- mandatory for test teams and architects:
    - the test team needs to know what the test case creation process is, what the life cycle of the test case is, and which elements the test case includes
    - architects must check whether the test cases are consistent with the (system) specifications
- recommended for developers:
    - developers should know what the life cycle of the test case is and how the system will be tested
- optional for the rest of the project members.

1.3 Structure of the document

This deliverable presents the following information contained in given chapters:

2- Testing methodology- description of test approach used during project

3- Installation guide- system requirements, installation procedure and configuration of the platform

4- Performed Test cases- contains list of Test Cases

5- Reported issues- contains list of issues reported in JIRA tool with statuses

6- Conclusions- contains summary of this document.

2 Testing methodology

Approach to provide quality to the project was based on ISTQB guidance (https://www.istqb.org) coordinated by personnel with Foundation Level certificate in this area. Software testing is performed to assess its quality and to reduce the risk of a failure in operation. The testing process consist of many stages such as planning, monitoring, creating, conducting the tests, reporting progress and results and final assessment. For every development activity should be a corresponding test activity. Software development process is based on Agile methodology that focuses on flexibility, collaboration and efficiency with iterative incremental approach, where planning, executing and evaluating requirements and solutions evolve through collaboration between self-organizing cross-functional teams. It helped us reduce time consuming blocking issues and create solutions for the new ones that appearing.

Core components of good Quality Assurance approach such as:

1. unit tests: test the building blocks of an application, typically in isolation from the application's other units and components, code level testing prepared and executed by development team
2. integration tests: to ensure that all interacting components are operating correctly together and each feature/functionality of the system can be successfully executed
3. functional tests: validates the features and operational behaviour of software to ensure that they correspond to its specifications, at least one separate test environment will be set up
4. regression tests: testing previous features of the system (not delivered in current release), to ensure that they're still working properly after a software change

helped us provide a high quality of the MORPHEMIC platform.

In the planning phase we decided what new components and functionalities would be introduced in the new release. Next, during the development phase, Unit and Integration testing were conducted by the respective development teams. After the developers finished implementing a new feature or a bugfix, they can open a merge request, where other team members can leave their comments. To improve the workflow, we used CI/CD pipeline on https://gitlab.ow2.org so, developers could make changes to the code and then test it and push it out for delivery and deployment. It helps to work simultaneously on the different components and merge changes into repository as often as possible.

The test process started with a creation of a Test Plan containing specific Test Cases for every functionality, feature and use case. A test should be carried out for each new element added in development stage and integrated into the system. To create a Test Case and to report any Bugs we used JIRA (D4.5- Test cases and testing).

2.1 Test Case creation

A test case is a set of test data, pre-conditions, expected results and post-conditions for tested components. A test cases describes how to perform a specific test and are created by the team of testers.

Test cases are prepared in JIRA https://www.atlassian.com/software/jira

- Open https://jira.7bulls.eu
- Enter your credentials in the fields *Username* and *Password* and press the *Log In* button
- To create a new test case, you need to choose *Create* tab

- Fill in all required fields: Project name, Issue Type, Summary (short description or title of the test case), Reporter (name of the tester that created the test case), choose Priority, Assignee (name of the tester that will perform the test), Input conditions (basic information about the platform that it's used), Steps to Complete, Expected Results
- After pressing the *Create* button a new Test Case is created.

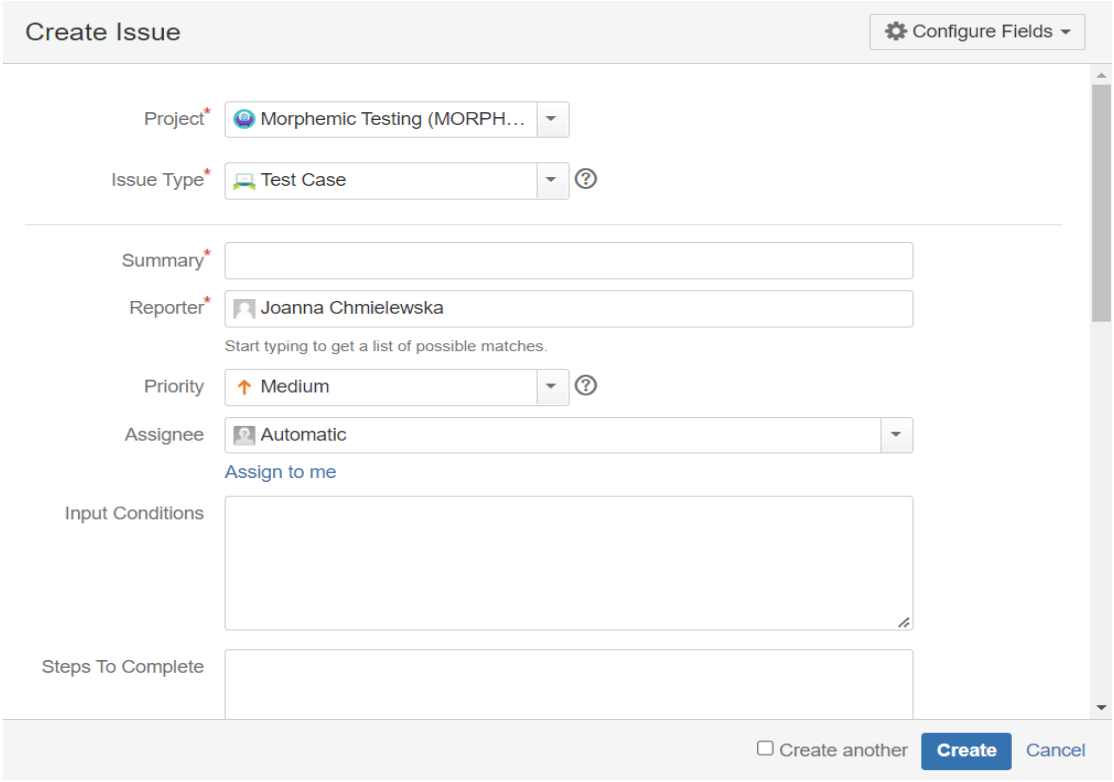

Figure 1 Creating a Test Case in JIRA

2.2 Life cycle of a test case

After we created the Test Case it has a status *New*. Next step is to accept it and change for *TO DO* and next for *IN PROGRESS* when a team member starting to work on it. If the Test Case is executed without any issues, we are changing the status for *TO TEST* and performing all the steps that were given in the description. After that step you can choose the status *DONE* (if the results are the same as it was expected) or *REOPEN* (if there are any problems). At the end of a cycle, you can choose a status *CLOSED* but if needed you can always *REOPEN* it.

After executing a Test Case there could be three possible outcomes:

- Passed- when system works according to the assumptions and received results are the same as expected results
- Failed- when system doesn't work or works differently than in the assumptions, received results are different than expected results related or a bug has been created to the test case
- Blocked- when one or more steps are blocked by another test case, one or more steps are blocked by a bug or part of system is blocked by another part of system which does not work.

Test cases with statuses 'Blocked' and 'Failed' were retested.

2.3 Test documentation

Every time a new feature or new functionality is added, the test documentation is needed so test team works together with the development teams of all participants, resulting in an improvement of test documentation, test cases and code.

The objective of the documents is to define the strategy that will be used to test the individual components and the integrated MORPHEMIC platform. A Test Plan documents all the stages of workflow to verify and ensure that all new features meet their specifications and requirements. It contains the details of all the processes, resources and goals for a specific test of the subsystem and component. The documentation explains what software need to be used in the tests and defines the criteria for passing or failing, also who will perform the tests and all the preconditions and the steps needed to be followed by the testing team.

3 Installation Guide

3.1 Technical Requirements

For the installation of the MORPHEMIC platform with version from release 2.5 we used 2 different installation setups:

1. Basic installation script has requirements based on r5.xlarge virtual machine from AWS resources we used in our process of testing and it includes 4 CPUs and 32 GB of RAM. We managed to discover that the biggest RAM usage is encountered during the installation process and further usage of the platform does not generate such load on both CPU and RAM. Recommended disk size is 100GB.
2. Second script creates 20 GB of Swap Space on the hard drive allocated to platform installation. This procedure allowed to decrease the requirement and use r5.large AWS virtual machine during our test which resulted in 2 CPUs and 16 GB of RAM with a cost of little more time for installation of the platform without changes in further use. Suggested disk space for this setup is 100GB where 80GB are allocated for the Morphemic server and 20GB to be used as a Swap memory.

Installation script with Swap Space from file installMorphemicSwapUnattended.sh available in repository https://confluence.7bulls.eu/display/MOR/%5BMAIN+INSTALLATION+DOC%5D%5BMORPHEMIC%5D+RC2.5+with+SWAP+platform+installation+guide

We need to expose some ports in the machine to allow inbound traffic reaching our services (Table 1).

Table 1 Accessible ports

Port	Protocol	Component	Purpose
22	TCP	ssh	Console
80	TCP	UI frontend	Melod UI frontend
443	TCP	UI frontend	Melod UI frontend SSL
8088	TCP	ESB	REST API
8095	TCP	Camunda UI	Process UI
8181 8998 7077 38000 38100 38200 38300 38400 38500	TCP	Spark	Spark components
61610-61619	TCP	EMS	ActiveMQ event broker ports
2222	TCP	EMS	Baguette server port
1099	TCP	EMS	ActiveMQ JMX connector port

8111	TCP	EMS	REST API of EMS
8078	TCP	UI backend	Melodic UI backend
2036	TCP	CDO Server	CDO Server
3077	TCP	JWT	JWT
2121 4433	TCP	webssh	webssh
3000	TCP	Grafana	
8123	TCP	mq-http-adapter/UI	(optional, if diagnosis endpoint is used)
8097	TCP	Adapter	
8880	TCP	Proactive Scheduler	
33647	TCP	Proactive Scheduler	

3.2 Recommended installation procedure

Since our target customers is a wide array of developers in different sectors, we aimed at making the server installation as automated as possible. This can be reflected in the installation steps:

1) First step is provisioning and accessing machine for MORPHEMIC installation and exposing the ports that are mentioned in the table above. In our case we based our tests mainly on VMs created on AWS resources. The first thing would be getting access to the VM using ssh connection
2) Then we need to access user's home directory (on AWS VM with ubuntu it would be /home/ubuntu)
3) Download the installation file by cloning the git repository

   git clone https://gitlab.ow2.org/melodic/melodic-utils.git

4) Checkout to the branch of release 2.5

   cd ~/melodic-utils

   and then

   git checkout morphemic-rc2.5

5) Run the installation script:

   sudo -E ~/melodic-utils/melodic_installation/installMorphemicSwapUnattended.sh

The installation will start by creating Swap Space, this takes some time and reports when it finished. After that the rest of the installation reports every step in actual time.

3.3 Basic configuration of MORPHEMIC platform

After installation some basic configuration should be performed to make the MORPHEMIC platform available for usage.

3.3.1 Loading system profile with aliases created for MORPHEMIC

To load profile created for MORPHEMIC and containing useful aliases that can be used to manage platform:

1. Go to the /home/ubuntu directory. Right after installation following command should be performed:

   cd ~/

2. *Source the profile file created during installation process:*

   . .profile

3.3.2 Starting MORPHEMIC platform

To start the docker containers and the integration between them use one of the aliases available in the previously loaded profile. To do this simply run:

drestart

To check if platform started correctly, please use alias:

mping

This alias lists docker containers used as part of the project with their current status, the platform has started properly if all of them are in OK status.

4 Performed test cases

This subsection presents a list of executed Test Cases for MORPHEMIC release 2.5. We could divide them into few groups:

- initial deployment- this group contains all scenarios related to the initial deployment of an application in the Melodic platform.
- metric management- collection, processing (aggregation), storage and delivery of raw and composite metrics, as well as CAMEL events based on these metrics
- application creation- test cases related to designing, creating and exporting a CAMEL model
- reconfiguration- reconfiguration of the application based on the new solution found by Reasoning part of the system
- forecasting module- related to the forecasting module

Table 2 shows the identifier of the executed test cases with a summary of each case.

Table 2 Test cases executed in MORPHEMIC release 2.5

KEY	Name
MORPHTEST-196	2.5 FCR with different reconfiguration options (proactive/reactive)
MORPHTEST-197	2.5 - Create Utility Function by function
MORPHTEST-198	2.5 - Create Utility Function by template
MORPHTEST-199	2.5 - Check NBeats forecaster results for Genom
MORPHTEST-200	2.5 - Check NBeats forecaster results for FCR
MORPHTEST-201	2.5 - Check CNN forecaster results for FCR
MORPHTEST-202	2.5 - Check CNN forecaster results for Genom
MORPHTEST-203	2.5 - Check Prophet forecaster results for FCR
MORPHTEST-204	2.5 - Check Prophet forecaster results for Genom
MORPHTEST-205	2.5 - Check Gluon forecaster results for FCR
MORPHTEST-206	2.5 - Check Gluon forecaster results for Genom
MORPHTEST-207	2.5 - Check eshybrd forecaster results for Genom
MORPHTEST-208	2.5 - Check Arima forecaster results for Genom
MORPHTEST-209	2.5 - Utility function creator: creation by function
MORPHTEST-210	2.5 - Utility function creator: creation by template
MORPHTEST-211	2.5 - Exponential Smoothing Predictor: Simple usage of the core prediction functionality
MORPHTEST-212	2.5 - SLO Severity-based Violation Detector
MORPHTEST-213	2.5 - Installation and deployment of FCR application on AWS
MORPHTEST-214	2.5 - Installation and deployment of two component application on Openstack
MORPHTEST-215	2.5 - Installation and deployment of Genom application on AWS
MORPHTEST-216	2.5 - Installation and deployment of Genom application on AWS
MORPHTEST-217	2.5 - Assure that wrong cloud configuration forbids retrieval of Node Candidates
MORPHTEST-218	2.5 - Morphemic - Deployment of a FCR application on one Cloud Provider on machine with artifacts from previous deploy
MORPHTEST-221	Deployment of a multipleByon TwoComponentApp on AWS
MORPHTEST-222	Deployment of an Edge DBComponentApp on AWS

5 Reported issues

During the development and testing of the system bugs can be found and reported in JIRA. Whenever an inconsistency was detected between actual and expected results it was logged in a bug report for further follow up. A bug is a defect in a component or system that can cause them not performing as they were required and cause failure. Sufficient details (system environment, steps to follow) to how the bug was detected should be provided for investigating the issue and to be able to recreate the process when it was first discovered and reproduce the problem. Over its entire lifecycle, a defect

changing status depending on the progress being made in resolving the issue. After defects/issues were fixed and the resolution was released, testing was conducted to verify if the resolution was successful and to ensure that the system is working correctly. The logging is performed directly in JIRA, which is the only official channel for inconsistency reporting for this project and it helps maintaining communication during resolving the issues.

Table 3 contains bugs reported and resolved within Release 2.5 with KEY, Summary and status

Table 3 Issues in Release 2.5

KEY	Name	Status
MOR-198	Problem with automatic installation of Proactive Scheduler	Closed
MOR-201	Eshyrid forecaster is not sending proper predictions	Closed
MOR-206	Ubuntu 18 offer not fetched	Closed
MOR-207	Openstack deployment: Bad login and preparInfra stage fail	Closed
MOR-208	Error while deploying the application in a private network.	Closed
MOR-212	exponential-smoothing does not build - pipeline fails	Closed
MOR-215	Genom fails due to jcloud and genom scripts sprint?	Closed
MOR-219	EMS fails to start on morphemic-rc2.5 instance	Closed
MOR-232	Due to changes of EMS configuration file changes EMS is not starting properly after new branch build	Closed
MOR-237	Nbeats prediction for FCR is not present in influxDB	Closed
MOR-238	Gluon prediction for FCR is not present in influxDB	Closed
MOR-239	Exponential smoothing has a problem with connection to broker	Closed
MOR-240	Metric for Eshybrid is not included in dashboard for FCR	To do
MOR-241	Monitors fail to start on byon node	Closed
MOR-243	Grafana dashboard for GENOM gluon number of cores querry has typo	Closed
MOR-244	Nbeats prediction missing for GENOM app	Closed
MOR-245	Eshybrid metrics prediction for GENOM are missing in DB	Closed
MOR-247	Error during commiting transation between scheduler and mariaDB	Closed
MOR-248	Duplicate entry 'BYON_NS_2...' for key 'PRIMARY' in multiple BYON deployment	Closed
MOR-249	Installed 2.5 not working correctly on OpenStack	Closed
MOR-250	No event types for GLOBAL grouping in EMS when deploying multipleBYON TwoComponentApp	Closed
MOR-251	Problem with CDO database on OpenStack	Closed
MOR-252	Edge deployment fails at Generating Constraint Problem stage	Closed
MOR-253	[ICON] RC3.0 proActive server seems not installed	Closed
MOR-254	[RC3.0] drestart shows errors	Closed
MOR-256	Error by getting constraint problem. Table doesn't exist.	Closed

MOR-257	docker-compose down does not work in morphemic subfolder	Closed
MOR-258	Not integrated components are reported by mping alias as NOK	Closed
MOR-259	error by deployment	Closed
MOR-261	Scheduler fails to deploy nodes on OpenStack resources	Closed
MOR-262	FCR deployment on Openstack results in background process not performed	new
MOR-263	[CHUV] RC2.5 Command purgesal throws ERROR 1451: Cannot delete or update a parent row	Closed
MOR-264	NullPointerException during connection to Resource Manager in multipleBYON deploy	Closed
MOR-266	Eshybrid metrics for WillFinishTooSoon are missing inDB	Closed
MOR-268	Wrong IP in ems logs.	To test
MOR-269	Sensitive Values are not waved into the install section of the component install section	New
MOR-271	2 Exponential smoothing metrics are not present in Grafana	New
MOR-273	ICON application problem NullPointerException: null at org.activeeon.morphemic.model.EmsDeploymentRequest. getWorkflowMap(EmsDeploymentRequest.java:162) in adapter log	Closed
MOR-278	remove free worker doesnt work	New

6 of reported bugs had highest priority, 6 had priority set to high, 17 issues had medium priority.

6 Conclusions

Our approach for release 2.5 allowed us to keep the same performance level, while at the same time lowering the cost of using the MORPHEMIC platform both for our further tests and usage of target users. We achieved this by not only adding possibility to use more nodes in BYON feature and adding possibility to use EDGE nodes as new feature, but by creating new installation script creating Swap Space lowering the platform's hardware requirements. Encountered complication connected to OpenStack resources allowed us to learn and prepare for further integration with different cloud providers.

Consortium agreed to change the quality assurance approach by creating basic test documentation during the deployment phase of adding new features. This approach gives the opportunity to keep Quality Assurance documentation with high quality and find possible problems at early stages of development which is crucial for reducing time spent on resolving issues.

Changes in the components described in the new structure complicated the integration process but were necessary to add new features with improvement of both possibilities and quality of the MORPHEMIC platform.

During tests of Release 2.5 39 issues were reported and 6 of them were moved for resolution in the next sprint. Those cases were discussed with reporters and were not blocking work of features planned for Release 2.5. All other issues were resolved and tested before changing their status to Closed.