MORPHEMIC

D6.1 Industrial requirements analysis

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 –
31 December 2022

www.MORPHEMIC.cloud

Deliverable reference
D6.1

Date
31 August 2020

Responsible partner
Engineering

Editor(s)
Ciro Formisano

Reviewers
Michał Semczuk, Yiannis Verginadis, Paweł Skrzypek, Alessandra Bagnato

Distribution
Public

Availability
www.MORPHEMIC.cloud

Executive summary

This deliverable defines, analyses and discusses the industrial requirements of the MORPHEMIC platform: each requirement will include also a fulfilment metric that will be the starting point of the validation framework. Part of the concepts and functionalities of MORPHEMIC are inherited from the MELODIC platform, on which the first part of the document (chapter 2) is focused. Moreover, MORPHEMIC takes into account new needs that technology and real-world applications nowadays should meet. Specifically, they need on one hand more and more resources, and, on the other hand, to support different models, such as Edge and Fog, besides the Cloud. Polymorphic and Proactive Modelling, Planning and Adaptation, basis of the MORPHEMIC platform, respond to these requests. Section 2.1 recaps the objectives of the MORPHEMIC project, briefly describing CAMEL language, used in MELODIC for modelling applications to be deployed on different Cloud infrastructures and Edge resources. The first two sets of requirements (section 2.4) come directly from MELODIC and from the analysis of the objectives and the preliminary architecture of MORPHEMIC proposed in the DoW. One of the core concepts of MELODIC that will be kept in MORPHEMIC is the concept of *Utility*: a value associated with the benefits that a certain application produces. The continuous maximization of the *Utility* is the reference on which decisions on application deployment or re-deployment will be taken.
The second part of the deliverable (chapter 3) is focused on the use-cases. They embrace a wide range of application areas, including software defined networks (section 3.1), healthcare (section 3.2) and computational fluid dynamics (section 3.3). Each use-case provides a set of requirements: some of them are common with the other use-cases and similar to the ones coming from the analysis of chapter 2, others are specific and provide more details on what should be designed. Chapter 4 summarizes the requirements found in the other sections and proposes two evaluations methodologies. The first one represents the *fulfilment metrics* of the requirements and will be valid for the whole project. It is based on a set of *fit criteria* that will define whether each requirement will be fulfilled or not by the platform that will be implemented. The second methodology aims at defining whether the contributions of the use-cases' requirement has modified, added or removed some functionalities of the original concept of MORPHEMIC. The result of this preliminary analysis is that the starting point, MELODIC, is correct and must be strongly improved during the project period to get the final goal.

Author(s)
Ciro Formisano (ENG), Robert Gdowski (IS-W), Adeliya Latypova (CHUV) Ferath Kherif (CHUV), Sebastian Geller (ICON)

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
24/06/2020	0.1 (draft)	ENG	First draft
02/07/2020	1.0	ENG	Preliminary version
2/7/2020	1.1	7BULLS	Review by Pawel Skrzypek
8/7/2020	1.2	NTUA	Review by Yiannis Verginadis
16/07/2020	1.3	7BULLS	Official review by Michal Semczuk
23/07/2020	2.0	ENG	Revised version
25/07/2020	2.1	SOFT	Official review by Alessandra Bagnato
27/07/2020	2.2	ISW	Modifications on the use-case 1 by Robert Gdowski
30/07/2020	3.0	ENG	Ready-for-release version
25/08/2020	3.1	ENG	Minor fixes by Ciro Formisano
27/10/2020	3.2	ENG	Re-submission after the review of the PMB
09/10/2020	3.3	ICON/ENG	Modifications on the third use-case
16/11/2020	3.4	7BULLS/ENG	Revised version after the extra review by Michal Semczuk

Table of Contents

Index of Figures

Index of Tables

Glossary

Acronyms	
AI	Artificial Intelligence
(S1) AP	(S1) Application Protocol
CAMEL	Cloud Application Model Language
CEP	Complex Event Processing
CFD	Computational Fluid Dynamics
CP	Control Plane
CPU	Central Processing Unit
CU	Central Unit
CU-CP	Central Unit – Control Plane
CU-UP	Central Unit – User Plane
DoW	Description of Work
DPDK	Data Plane Development Kit
DSL	Domain Specific Language
DU	Data Unit
FCAPS	Fault-management, configuration, accounting, performance and security
FPGA	Field Programmable Gate Array
GDPR	General Data Protection Regulation
HPC	High Performance Computing
IP	Internet Protocol
MAC	Media Access Control
ML	Machine Learning
MRI	Magnetic Resonance Imaging
NAAS	Network as a Service
NAS	Non-Access Stratum
NFV	Network Function Virtualization
PDCP	Packet Data Convergence Protocol
PNF	Physical Network Function
RAN	Radio Access Network
RF	Radio Frequency
RLC	Radio Link Control
ROM	Reduced Order Model
RRC	Radio Resource Control
RU	Radio Unit
SaaS	Software as a Service
SCTP	Stream Control Transmission Protocol
SDN	Software Defined Network
SD-RAN	Software Defined - RAN
SLA	Service Level Agreement
SPM	Statistical Parametric Mapping
UI	User Interface
UP	User Plane
VM	Virtual Machine
VNF	Virtualized Network Function

1 Introduction

The core of the MORPHEMIC platform[1] is the MELODIC project[2]. MELODIC is a *Multi-Cloud Optimization and adaptation platform* for generic cloud applications: it was designed and prototyped during a successful EU project and now has taken the form of a consolidated product. MELODIC introduces the concept of *deployment configuration* to define the deployment details of a certain application. The *multi-cloud* deployment is continuously adapted and kept up-to-date according to the desired QoS defined by the DevOps. MORPHEMIC, besides this, introduces the concepts of *polymorphic* and *proactive* modelling, planning and adaptation. "***Polymorphic modelling** proposes a unique way of **adapting** and optimizing Cloud applications in distributed environments combining various Cloud levels including Cloud data centres, edge Clouds, 5G base stations, and fog devices. On top of that,* MORPHEMIC *will support **Proactive Adaptation** not only based on the current execution context and conditions but also forecasting future resource needs and possible deployment configurations*"[3] .

The exposed concepts, the ones coming from MELODIC and the new ones introduced by MORPHEMIC, are the basis of the functionalities and the components that will be defined, designed and implemented during the project. A further remarkable concept originated in MELODIC will be reused and, probably, updated in MORPHEMIC: *Utility*. It represents the main decision element for each change introduced in the environment and application form. Utility is obtained through a *Utility Function* evaluated by the *Utility Generator* (part of the MELODIC architecture). The Utility is evaluated by processing two elements:

- the *business objectives* of the application
- the costs introduced by the *constraints* related to the requested Service Level Agreements (SLA) and by the elements required to deploy the application.

One of the objectives of the introduction of the MORPHEMIC platform in application deployments will be to keep the present and future Utilities of deployed applications as high as possible.

The requirements listed and analysed in this deliverable are in part based on the aforementioned aspects. Specifically, a first set of requirements the requirements are directly derived from MELODIC: they are listed in section 2.2, along with a brief description of the MELODIC platform. These requirements are enriched by a second set originated by the analysis of the new concepts introduced in MORPHEMIC and by the experience of the partners in managing applications in different forms and environments. This group of requirements is described in section 2.5, along with a preliminary analysis of the MORPHEMIC platform.
However, most of the work is focused on the use-cases. They were chosen to cover a wide application domain: from software-defined networking to industry through healthcare. Each domain has its specific issues and requirements, which the MORPHEMIC platform will have to meet. Section 3 contains the descriptions of each use-case, the benefits expected from MORPHEMIC and the related requirements.
Finally, section 4 contains the summary of all the requirements and the fulfilment metrics. Specifically, a *fit criterion* will be proposed to have a reference for verifying each requirement. A preliminary match between groups of requirements and the architectural elements proposed in the DoW will be attempted. The objective of this activity is to get at this stage any requirement not fully aligned with the functionalities identified in the DoW. These requirements could become valid proposals to extend the original idea of MORPHEMIC in a direction considered useful by the use-cases.
This deliverable is due on M8: at time of writing it, some of the functionalities to be provided are not fully defined yet. This is especially true for the requirements related to the use-cases: some of them will be detailed and finalized at the end of Y1 (Deliverable 6.3). For this reason, some new requirements may be defined and some old may be re-defined during the next months.

[1] http://MORPHEMIC.cloud/
[2] https://MELODIC.cloud/
[3] MORPHEMIC DoA

1.1 Structure of the document

This deliverable can be divided in two parts: the first one is focused on the analysis of the requirements and the second one concerns their preliminary validation.
In details:

- Requirements analysis:
  - chapter 2, including the requirements coming from MELODIC (sections 2.2 and 2.3) and from the objectives of the MORPHEMIC project (section 2.4)
  - chapter 3, including the requirements provided by the three use-cases (sections 3.1, 3.2 and 3.3)
- Preliminary validation:
  - chapter 4, including the fit criteria (section 4.1) and the comparison against the *tentative of architecture* defined in the DoW (section 4.2).

The requirements are collected in tables: apart from the ones coming from MELODIC (Table 1), whose evaluation is out of the scope of this document, all the requirements are associated to a topic. The *topics* are a tentative of classification against the main activities planned for the project. Specifically, the topics are:

- **Adaptation** (proactive or polymorphic), section 2.4
- **Application Modelling**, capability to retrieve the characteristic of a certain application starting from the functionalities and the code
- **Optimization**, capability to obtain the best possible result (where "best" is defined according to specific evaluation parameters) by using the minimum possible resources
- **Parameters for the Utility Function**, defining the parameters that will be used to evaluate "*Utility*" (section 2.3)
- **Security**, intended as security of the Platform and capability to offer a certain degree of security in the deployed applications
- **Self-Healing**, concerning the capability to detect violation of service level objectives and any dangerous situation concerning the deployed components, in order to enact new adaptations to solve them
- **Supported Application Forms**, the application forms useful for the use-cases (i.e., serverless, containers...)
- **Supported Environments**, the environments useful for the use-cases (i.e., cloud, edge...).

The requirements not coming from MELODIC are also prioritized according to the MoSCoW Model. The MoSCoW Model is based on 4 priority levels: **Must** have, **Should** have, **Could** have and **Won't** have referred to each requirement. It provides a quick and immediate reference to define the number of resources to be allocated. The main benefit of using this Model is the possibility to assign a specific percentage of resources to each category. Objective of the MORPHEMIC project is to produce a prototype as more useful as possible for the use-cases: for this reason, a simple and effective prioritization schema will allow to optimise the effort to provide benefits for the Use-Cases partners already with the first releases. MoSCoW model is also understandable for non-technical people: this will be useful to give a clear idea about which will provide MORPHEMIC by reading this deliverable.

Finally, chapter 5 summarizes the contents of the document and proposes some of the next steps.

1.2 Target Audience

According with the DoW, this deliverable is a public document providing information on MORPHEMIC. Specifically, most of the document contains the descriptions of the three Use-Cases and the related requirements. This information can be useful for potential Use-Cases that could get benefits from the MORPHEMIC platform. The sections on the architecture, even if the descriptions are high level, are aimed at a technical audience. The document also includes an overall description of the MELODIC platform providing the basic information to get the starting point of MORPHEMIC.

1.3 Requirements encoding

The requirements included in this deliverable are identified through a unique ID. Each ID defines the origin of the requirement:

- the requirements coming from MELODIC (Table 1 and Table 2) are identified by an id with the format **MEL-XX** and **MEL-NF-XX** (for non-functional requirements)
- the requirements coming from the MORPHEMIC's initial idea (Table 3) are identified as **MOR-<topic>.XX**
- the requirements common to the three use-cases are identified as **UC-C-<topic>.XX**
- the requirements of each use-case are identified as **UC-<use-case number>-<topic>.XX**.

Where <*topic*> identifies a *fine-grained categorization* that helps to identify each requirement, specifically

- **AD** Adaptation
- **MD** Application Modelling
- **OP** Optimization
- **UF** Parameters for the Utility Function
- **SEC** Security
- **SH** Self-Healing
- **SA** Supported Application Form
- **SE** Supported Environments.

Finally, *XX* represent a progressive number associated to the specific requirement.

2 The MORPHEMIC platform

"*MORPHEMIC proposes a unique way of adapting and optimizing Cloud computing applications by introducing the novel concepts of polymorph architecture and Proactive Adaptation*".[4] Most of the concepts considered important by the use-cases partners are included in this sentence, in particular:

- *adaptation* and *optimization*, to adapt the application configuration (i.e., quantity and quality of resources) so as to optimise the application utility according to the current context
- *polymorphic architecture*, to allow heterogeneous application components to be hosted on heterogeneous hosting resources (Docker, bare metal, virtual machines…) and services, according to the specific requirements and context
- *Proactive Adaptation*, to forecast the future needs of the deployed application and to fulfil them by providing the required resources at the right time.

This chapter aims at connecting the MELODIC project, the starting point, with the first idea of the MORPHEMIC platform, defined in the description of work mainly by the experience of the involved partners. Specifically, section 2.1 summarizes the objectives of MORPHEMIC to better define the scope and the target of the project; sections 2.2 and 2.3 provide descriptions of the MELODIC platform and of the concept of *Utility*. Finally, sections 2.4 and 2.5 are focused on MORPHEMIC, providing the first requirements and the preliminary architecture according to the descriptions of the DoW.

2.1 Objectives

The technical objectives of MORPHEMIC, listed in the project documentation, are the following:

1. *to extend a Cloud application modelling language with polymorphic capabilities*
2. *to predict confidently the application behaviour*
3. *to predict and model adaptive Utility*
4. *to morph the model for optimized deployment*
5. *to provide an application lifecycle operational environment.*

Cloud Application Model Language (CAMEL[5]) is a modelling language that "*enables the management of self-adaptive cross-cloud applications (i.e., cross-cloud applications that autonomously adapt to changes in the environment, requirements, and usage)*"[1]. This language, proposed in PaaSage EU project[6] and improved in MELODIC, will be *extended with polymorphic capabilities*, as described in objective 1. Specifically, the concept of *polymorphed application in time*, introduced in MORPHEMIC, means that an application could be *polymorphed* in time to better adapt to changing conditions during its provisioning, including not only the present but also the forecasted future situations (objective 2). This means that MORPHEMIC will overcome the concept of fixed deployment configuration, enabling to change application form (container, serverless, VMs…) and provisioning environment (cloud, fog, edge…) in real time, basing on current and predicted application behaviour to maximize benefits for the application's user (objectives 3 and 4).
This could require to extend some components of MELODIC before the integration with the new, aforementioned, functionalities. The final result of this integration will be an *application lifecycle operational environment* as stated in objective 5.

The five objectives directly define part of the requirements which represent the starting point of MORPHEMIC. In this sense, the requirements are *objectives* by themselves to be expressed as fractal, by defining, in a detailed way the aforementioned objectives. The resulting list will be integrated and extended through the *industrial* requirements

[4] MORPHEMIC DoA
[5] http://camel-dsl.org/
[6] https://paasage.ercim.eu/

provided by the use-case partners. Specifically, the use-case partners present three applications having some *expectations* from MORPHEMIC. These expectations are the use-case needs and objectives on which a second list of requirements is produced: it is important that this list is fully compliant with the previous one. This will assure also that the use-case partners have clear ideas on the benefits that MORPHEMIC will provide to their businesses and that technological partners and use-case partners will easily proceed *on the same page* for the whole project. Finally, since MORPHEMIC extends MELODIC, a brief description of the MELODIC platform including its original requirements is useful to completely define the new platform. The next section is dedicated to that.

2.2 The MELODIC platform

This section is a summary of the deliverable 2.2 of the MELODIC project [2]: please, refer to that document for more information[7].

The MELODIC platform allows "*applications to run within defined security, cost, and performance boundaries seamlessly on geographically distributed and federated cloud infrastructures*"[8]. In other words, the application is *cloud agnostic*. The selection of cloud providers and cloud resources is fully optimized based on different parameters, including prices, performance and reliability. After the initial deployment the application is continuously monitored for checking the business goals fulfilment and appropriately reconfigured to always preserve an optimal service level.
Figure 1 shows the architecture of the MELODIC platform. It is conceptually divided into three main component groups, the interfaces to the end users, the *Upperware*, and the *Executionware*. The interfaces to the end users include tools used to model applications and datasets and interact with the rest of the platform. The MELODIC's modelling interfaces, through the CAMEL modelling language, provides a rich set of domain-specific languages (DSLs) which covers different modelling aspects, spanning both the design and the runtime of a Cloud application as well as data modelling traits. Applications and data models created through the modelling interfaces, in the form of CAMEL, are given as input to the Upperware. The job of the Upperware is to calculate the optimal data placements and application deployments on dynamically acquired Cross-Cloud resources in accordance to the specified application and data models in CAMEL as well as in consideration of the current Cloud performance, workload situation, and costs. The actual Cloud deployments are carried out through the Executionware. The Executionware is capable of managing and orchestrating diverse Cloud resources, and it also supports the cross-cloud monitoring of the deployed applications. Besides the three main component groups, two auxiliary services have also been implemented: one of them enables a unified and integrated event notification mechanism, the second one warrants secure operations with the MELODIC platform.

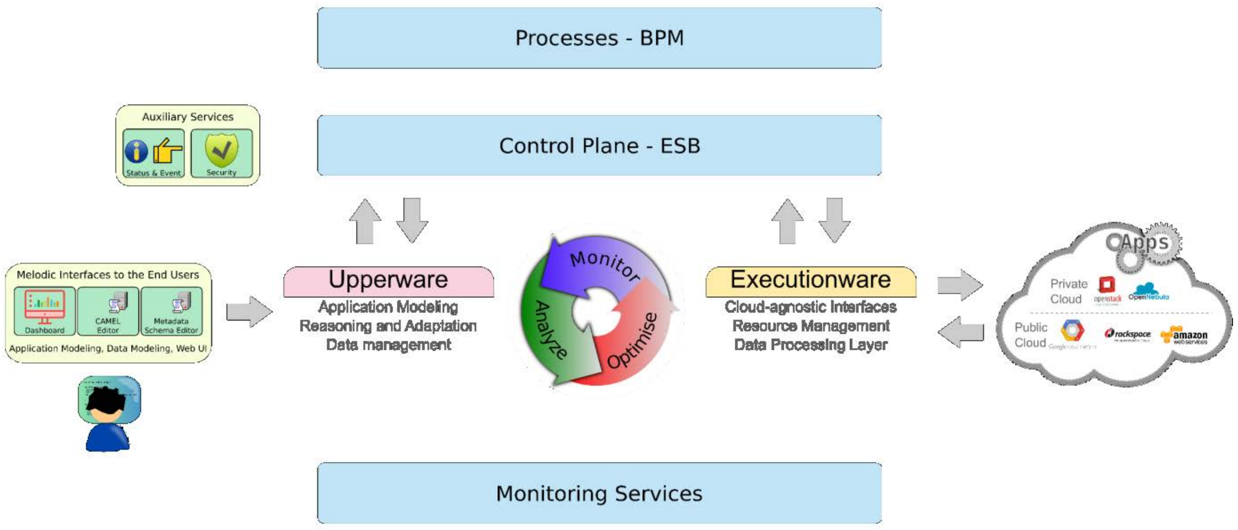


Figure 1 MELODIC Architecture

[7] https://h2020.MELODIC.cloud/deliverables/
[8] https://cordis.europa.eu/project/id/731664

MELODIC provides a certain degree of dynamicity to deployment configuration. This happens through the Utility Function (section 2.3), which, according to the set parameters, produces the utility values to be used to modify the configuration.

2.3 The concept of Utility

MELODIC allows modifying the *deployment configurations* according to some requirements related to the deployed application and the context. The requirements used should be formalised as much as possible to be correctly adopted as input for the MELODIC's decision-making. The concept of application *Utility* helps to achieve this. The Utility is obtained by processing some *parameters* based on the aforementioned set of requirements. The main concepts from which the analysis should start are:

- the *business goals* of the application, which represent the high-level motivation of the deployment
- the *constraints* mainly related to costs or Service Level Agreements (SLA) with the users that provide limits to the usable resources.

Generally speaking, for business goals and costs (directly or indirectly related to the constraints), the requirements are very simple:

- maximize incomings
- minimize costs.

These requirements represent, for the Application Manager, targets to beget: application-specific parameters can be used to define these targets. Usually, these parameters include:

- Budget
- Number of Users
- Response Time
- Memory Usage
- Number of Transactions per minute.

Each of these parameters can have a certain threshold, related to the business goals and the constraints. The thresholds are mandatory, but under the thresholds it is possible to modify all the values to obtain the best trade-off at a certain moment and in a certain context. The evaluation of the best trade-off is the process aimed at maximizing the *Utility* of the application.
The *Utility Function* of a certain application gets all the associated parameters as input and provides the combination to maximize the *Utility value*, that is represented as a number between 0 and 1.

Autonomic Cross-Cloud application management, on which MELODIC is based, is founded on Utility value and Function and on the possibility to dynamically change the deployment configuration to maximize the Utility value.
In MELODIC this process is performed through a *Utility Generator* in the way exposed in Figure 2.

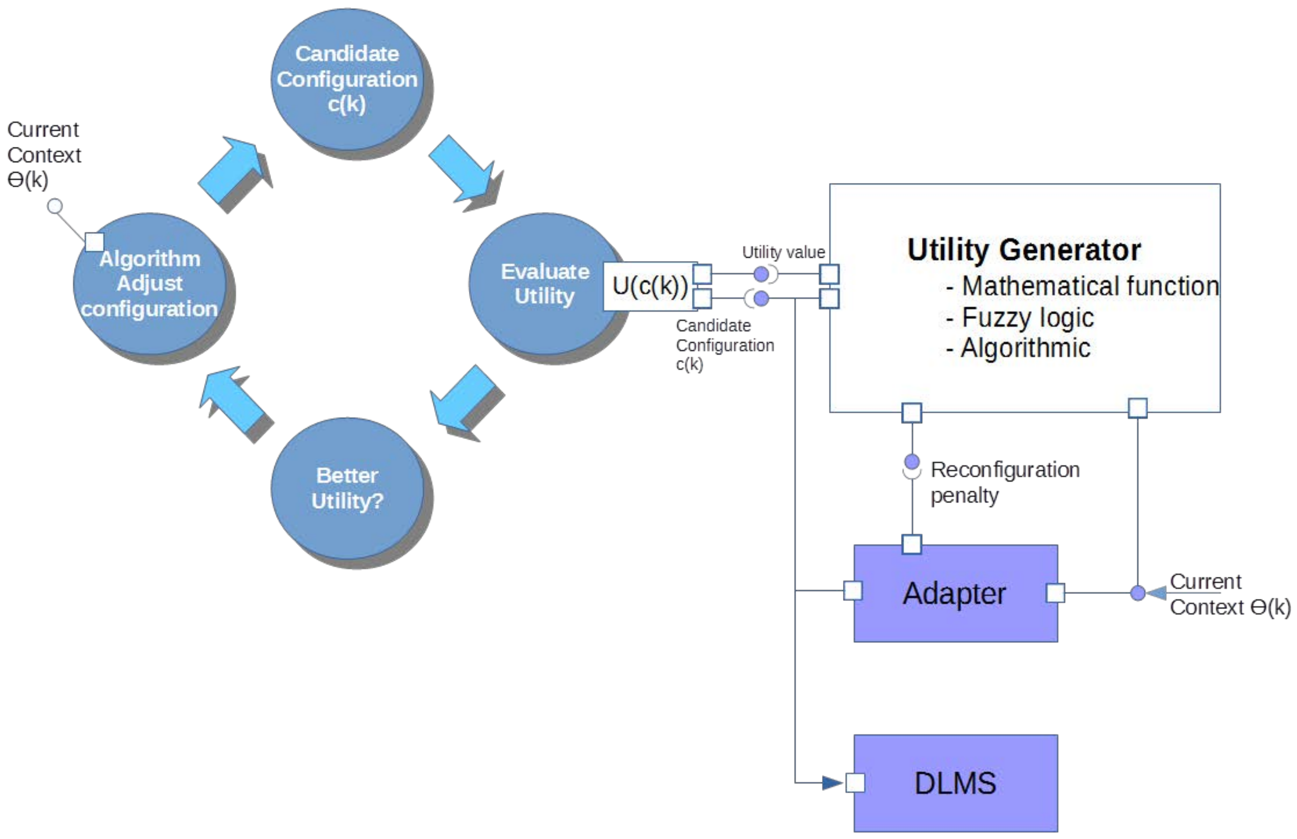


Figure 2 MELODIC Utility generator

2.4 Requirements and functionalities of MORPHEMIC

MORPHEMIC will not be designed from scratch. The basic concepts of MELODIC (section 2.2) have been extended according to the objectives listed in section 2.1. Starting from these considerations it is possible to define a first set of requirements describing the goals and the idea of MORPHEMIC according to what has been proposed and the experience of the partners.
Specifically, these requirements can be divided into two groups: the first one contains the requirements of MELODIC, the second one contains the new requirements introduced in MORPHEMIC. The MELODIC's requirements, shown in Table 1, are described in the deliverable 2.1[3] of the MELODIC project. They represent the goals of MELODIC and give an idea of the functionalities: the evaluation about how much they have been satisfied during the MELODIC project is out of the scope of this document. However, the presented list is useful to understand which is the starting point of MORPHEMIC.

Table 1 Functional requirements coming from MELODIC

ID	Name	Description
MEL-1	Automated and transparent deployment of data-intensive applications	MORPHEMIC must find the best possible combination of Cloud services to deploy an application according to the requirements, in order to limit the user intervention on this aspect
MEL-2	Ability to allow platform control flow to wait for the user instructions (through a user interface), if required	The user interface must be interactive in order to enable the user to set the commands if and when required
MEL-3	A generic metadata structure complementing unified data management, access control, and data-aware application	MORPHEMIC must provide a Data management approach in order to support data deployment on different environments

	design	
MEL-4	Efficient data placement, migration, and post-migration methodologies and algorithms for Multi-Clouds	MORPHEMIC must provide support for efficient data migration (if and when required) from an environment to another one
MEL-5	Support for a large variety of structured, unstructured, and hybrid data sources in heterogeneous environments	MORPHEMIC must provide support for different data sources
MEL-6	Capabilities to continuously monitor deployed applications	MORPHEMIC must be able to monitor deployed applications in order to efficiently manage multi-cloud adaptation
MEL-7	Support for model@runtime for dynamic runtime adaptation of deployed applications	MORPHEMIC must provide support for run-time adaptation of multi-cloud user applications. The parameters that must be taken into account include variable Cloud efficiency in real-time, changes in end-user data processing requirements, resource reconfigurations together with alternative application deployment in order to sustain the service level demanded by the application provider
MEL-8	Secure and context-aware data access control mechanism	MORPHEMIC must offer the appropriate tools that guarantee the secure access and processing of all the involved data, handled by big-data intensive applications deployed and maintained in Multi-Clouds: any user should be properly authenticated before accessing data that reside in Cloud infrastructures.
MEL-9	Ability to allow for user-defined data security and confidentiality requirements	The data owner should be able to indicate as a requirement the need for cryptographically protecting her data that may be transferred, processed, and stored over public or private clouds
MEL-10	Data-aware scheduling of big data applications	MORPHEMIC should be able to consider data to take decisions on application scheduling
MEL-11	Efficient use of resources on the private infrastructures	MORPHEMIC must be able to include the knowledge of private infrastructures into the reasoning process, allowing the concrete and more optimal mapping of applications to specific physical nodes in the private Cloud
MEL-12	Intelligent mapping of application components on to the actual hardware	MORPHEMIC must be able to discover physical nodes, their meta-data, such as hardware configuration, and the network topology of the private Cloud
MEL-13	Support for computational and data scaling	MORPHEMIC must be able to acquire more resources when needed and release them when not needed in order to appropriately handle the workload

Part of these, are generalised requirements of Multi-Cloud and Cross-Cloud deployments and executions of data intensive applications. Another part come from the use-cases of the MELODIC project: the result is a complete and consistent platform that will be extended in MORPHEMIC.
In is important to notice that requirements MEL-8 and MEL-9 indicate that mechanisms of Access Controls and User Management have been included in MELODIC: they will be present in MORPHEMIC as well.
Table 2 contains the non-functional requirements that will be present also in MORPHEMIC exactly in the same form.

Table 2 MELODIC non-functional requirements

ID	Name	Description
MEL-NF-1	Extensibility	MORPHEMIC should consider future extensions. In other terms it should be possible to expand the system and the provided services with a reasonable effort
MEL-NF-2	Reusability	MORPHEMIC must be able to re-use existing assets in the

		software development lifecycle. Such assets can be products or by-product of this lifecycle and might include code, software components, designs and documentation
MEL-NF-3	Documentation	MORPHEMIC must be well documented
MEL-NF-4	Quality	Software functional and non-functional quality of MORPHEMIC must be high. Software functional quality prescribes how well the software complies with a certain design and the respective set of functional requirements and specifications. Software non-functional quality prescribes how well the software satisfies the non-functional requirements posed
MEL-NF-5	Fault Tolerance	MORPHEMIC must be able to continue operation even if one or more of its components have failed.
MEL-NF-6	Scalability	MORPHEMIC must be able to handle increased load with a proportional increase in the available system resources

The MELODIC's requirements will be *extended* in two directions and aligned with the use-cases proposed in MORPHEMIC. The directions in which the MELODIC's requirements will be extended are:

"***Polymorphic modelling*** *proposes a unique way of* ***adapting*** *and optimizing Cloud applications in distributed environments combining various Cloud levels including Cloud data centres, edge Clouds, 5G base stations, and fog devices. On top of that, MORPHEMIC will support* ***Proactive Adaptation*** *not only based on the current execution context and conditions but also forecasting future resource needs and possible deployment configurations*

1. **polymorphic** *planning*, *modelling* and **adaptation**, i.e., support for different *application forms* (container, serverless, VMs…) and provisioning environments (cloud, fog, edge…), non-only cloud infrastructures
2. **proactive** *planning*, *modelling* and **adaptation**, i.e., possibility to *forecast* the future utilities as well as evaluate in advance and apply, when requested, the appropriate modifications in the deployment configuration; not only decide basing on what has happened.

MORPHEMIC is able to predict the future *Utility* requested by a deployed application and to **proactively** modify the deployment configuration before the forecasted needs become actual. Furthermore, this adaptation will not be limited to cloud infrastructure but will support several environments.

Table 3 shows the specific requirements of MORPHEMIC. They come out from the original idea of the project, specifically from the objectives listed in section 2.1, and from the experience of the partners on application deployment. Each requirement is categorized for topic, as described in section 1.1. Moreover, since the prioritization and the evaluation will be very important for the effectiveness of the project, each requirement will have an associated MoSCoW priority.

Table 3 MORPHEMIC specific requirements

ID	Name	Description	Topic	Priority (MoSCoW)
MOR-SE.1	Polymorphic Environments: Cloud	MORPHEMIC must support Cloud	Supported Environments	MUST
MOR-SE.2	Polymorphic Environments: Hybrid Clouds	MORPHEMIC must support Hybrid Clouds	Supported Environments	MUST
MOR-SE.3	Polymorphic Environments: Multi-Cloud	MORPHEMIC must support Multi-Cloud environment	Supported Environments	MUST
MOR-SE.4	Polymorphic	MORPHEMIC could support Fog	Supported	COULD

	Environments: Fog		Environments	
MOR-SE.5	Polymorphic Environments: Edge	MORPHEMIC must support Edge	Supported Environments	MUST
MOR-SE.6	Polymorphic Environments: bare metal	MORPHEMIC could support on-premises environment	Supported Environments	COULD
MOR-SE.7	Polymorphic Environments: HPC	MORPHEMIC must support HPC	Supported Environments	MUST
MOR-SE.8	Polymorphic Environments: hardware accelerators	MORPHEMIC must support hardware accelerators	Supported Environments	MUST
MOR-SE.9	Polymorphic Environments: FPGA	MORPHEMIC must support FPGA	Supported Environments	MUST
MOR-SE.10	More polymorphic environments	MORPHEMIC could support environments different than the ones listed in MOR-SE.1-MOR-SE. 9	Supported Environments	COULD
MOR-SA.1	Polymorphic application forms: VM	MORPHEMIC must support Virtual Machines	Supported Application Forms	MUST
MOR-SA.2	Polymorphic application forms: containers	MORPHEMIC must support containers	Supported Application Forms	MUST
MOR-SA.3	Polymorphic application forms: serverless	MORPHEMIC must support serverless application form	Supported Application Forms	MUST
MOR-SA.4	More polymorphic application forms	MORPHEMIC could support application form different than the ones listed in MOR-SA.1-MOR-SA.3	Supported Application Forms	COULD
MOR-CON.1	Pre-configure multiple deployment configurations	MORPHEMIC must support multiple deployment configurations in order to plan the deployment for different contexts	Adaptation	MUST
MOR-SH.1	Real time infrastructure performance monitoring	MORPHEMIC must be able to monitor in real time the health status of the infrastructure, in order to use the obtained data for Proactive Adaptation	Self-Healing	MUST
MOR-SH.2	Real time applications performance monitoring	MORPHEMIC should be able to monitor in real time the health status of the applications, in order to use the obtained data for Proactive Adaptation	Self-Healing	SHOULD
MOR-SH.3	Self-Healing mechanism	MORPHEMIC must support a Self-Healing mechanism able to detect the violation of service level objectives or any other dangerous situations, concerning the deployed application components, in order to enact new adaptations.	Self-Healing	MUST
MOR-AD.1	Proactive Adaptation	MORPHEMIC must be able to adapt the deployment configuration (environment and application form) according to the predicted needs	Adaptation	MUST
MOR-AD.2	Prediction capabilities on applications	MORPHEMIC should be able to predict potential problematic trends on applications	Adaptation	SHOULD

MOR-AD.3	Prediction capabilities on infrastructures	MORPHEMIC should be able to predict potential problematic trends on infrastructures	Adaptation	SHOULD
MOR-MD.1	Application crawling	MORPHEMIC should be able to look for open-source software on a set of popular software repositories (such as GitHub, Apache...)	Application Modelling	SHOULD
MOR-MD.2	Application profiling	MORPHEMIC could be able to define application profiles according to patterns found on open-source software classified by functionalities	Application Modelling	COULD
MOR-OP.1	Optimization of Resources	Capability to reuse and optimize the usage of cloud resources as much as possible in the context of application reconfiguration	Optimization	SHOULD

MORPHEMIC will present a User Interface extending the MELODIC's one. The UI will meet a rich set of requirements analysed in WP 5 and described in the Deliverable 5.1, edited in parallel with the deliverable 6.1 and due at the same month (M8). The analysis of the UI requirements was performed in a way similar to the generic requirements: part of them comes from the concept of MORPHEMIC presented in the DoW and from the experience of the core partners, part comes from the requests of the use-cases.

2.5 Features of the MORPHEMIC platform

Figure 3 MORPHEMIC architecture tentative shows the first draft of the architecture of MORPHEMIC. The core of the architecture is MELODIC: its application modelling (CAMEL), profiling and reasoning capabilities will be used, along with the cloud agnostic deployment and functional capabilities. The polymorphic capabilities will be added through the new MORPHEMIC pre-processor element. Furthermore, the MELODIC's EMS will be extended to add advanced prediction and forecasting capabilities based on the state-of-the-art hybrid forecasting methods. Finally, the MELODIC's Utility Generator will be extended to cover Proactive Adaptation with time and future dimensions.

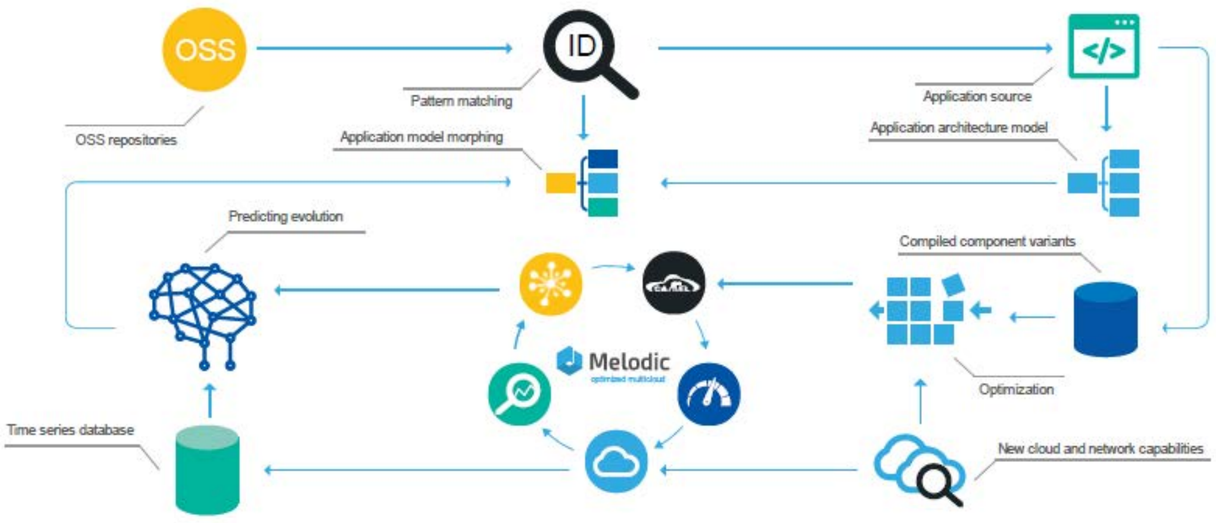


Figure 3 MORPHEMIC architecture tentative

Concerning the Utility Generator, the component is strictly related to the concept of *Utility Function*, defined in section 2.3. The concept will be preserved: this means that, as in MELODIC the *maximum Utility value* is the reference used for continuously optimize the deployment configuration of the applications, in MORPHEMIC an extended *Utility value*, including also references to Proactive Adaptation, will be used to continuously optimize the deployment and to plan the future optimized deployments basing on the forecasted contexts.

The main added-value features provided by MORPHEMIC are:

- **Polymorphic** and **Proactive** planning, modelling and **adaptation** (see section 2.4)
- **Self-Healing Capability**, associated with a monitoring system, to be resilient to failures or modifications of the external context
- **Hardware Accelerator Support** for applications deployed on the Cloud, in order to get flexibility and high performance
- **Uniform User Interface**, providing easy and consistent access to all the MORPHEMIC's tools.

Each use-case expects part (or all) of these benefits from MORPHEMIC. The tables included in the sections 3.1.2, 3.2.2 and 3.3.2 give the details for each use-case. The Table 4 summarises these benefits.

Table 4 Summary of the benefits expected by the three use-cases

Features	Use-case 1	Use-case 2	Use-case 3
Polymorphic Adaptation	✓	✓	✓
Proactive Adaptation	✓	✓	
Self-Healing Capability	✓	✓	✓
Hardware Accelerator Support	✓	✓	✓
Uniform User Interface	✓	✓	
Polymorphic Adaptation	✓	✓	

3 Use-cases Descriptions and Requirements

This section is focused on the three use-cases that will better define and evaluate the MORPHEMIC platform. The three use-cases belong to different application domains covering a wide range of usage possibilities. The use-cases will give an important contribution to consolidate and integrate the requirements obtained in chapter 2 from MELODIC and from the MORPHEMIC's initial idea.
The use-cases considered are:

1. *Virtualized base station for 5G cloud-RAN*, proposed by **IS-Wireless**. IS-Wireless is leader in the development of algorithms, protocols and tools for 4G and 5G mobile networks. IS-Wireless provides licensable, NFV-compatible (*Network Function Virtualization*) software implementing standard-compliant RAN (Radio Access Network) protocols ready for evolution to 5G.
2. *E-Brain Science*, proposed by **Centre Hospitalier Universitaire Vadois** (**CHUV**). CHUV is one of the five Swiss University hospitals. Specifically, the use-case can be applied to the work of the *Laboratoire de recherche en neuroimagerie* (LREN), which consists of a cross-disciplinary team of basic and clinical neuroscientists with an interest in the role of human brain structure and function in neurological disorders and healthy aging.
3. *Computational Fluid Dynamic Simulation*, proposed by **ICON**. ICON Technology & Process Consulting Limited operates in the high-tech field of Computational Fluid Dynamics (CFD) and provides blue-chip multi-sector engineering companies, their suppliers and consultants with the ability to predict fluid flow using 3D computer simulation.

Each subsection of this chapter is dedicated to a use-case. These sections have identical structures:

- sections 3.x.1 contain detailed descriptions, including the technological aspects known at time of writing
- sections 3.x.2 specify the features for which MORPHEMIC could provide benefits to each use-case
- sections 3.x.3 contain the functional and non-functional *requirements of MORPHEMIC provided by each use-case* (will be better specified below)
- sections 3.x.4 contain some technological requirements, focused on the resources (computing power, storage and network capabilities) needed for each use-case, if known: this is also useful to give an idea of the needed elasticity related to the external context (i.e., peaks of requests).

As mentioned above, the requirements of sections 3.x.3 are *requirements of MORPHEMIC provided by each use-case*: this means that, they are not requirements of the use-case, but requirements that *MORPHEMIC must meet to correctly support each use-case*. In other words, these requirements must be meet by MORPHEMIC in order to enable each use-case to meet the corresponding use-case-specific requirements. For instance, in order to enable a certain use-case to *support traffic isolation*, i.e.:

the application must support network traffic isolation in order to increase security

MORPHEMIC must:

be able to define deployment configurations supporting traffic isolation

The former is focused on the functionality to be provided and is use-case specific, the latter on the capability that MORPHEMIC must expose and is not specific of a certain use-case (even if has been suggested by it). As mentioned, these requirements can confirm the ones of chapter 2 or suggest new useful functionalities which must be taken into account to make the final outcome of this project an added value for users.

3.1 Virtualized base station for 5G cloud-RAN

This use-case, proposed by IS-Wireless[9], is focused on 5G Software defined Radio Access Networks. Specifically, MORPHEMIC will support the deployment of RAN building blocks on different infrastructures in an optimized way.

3.1.1 Use-case Description

The use-case is focused on 5G networks, in particular software solutions for RAN. Radio Access Network is the most expensive, cross-disciplinary and challenging part of a telecom network. Specifically, the capacity of the network is strongly dependent on the density of RAN network.

[9] https://www.is-wireless.com

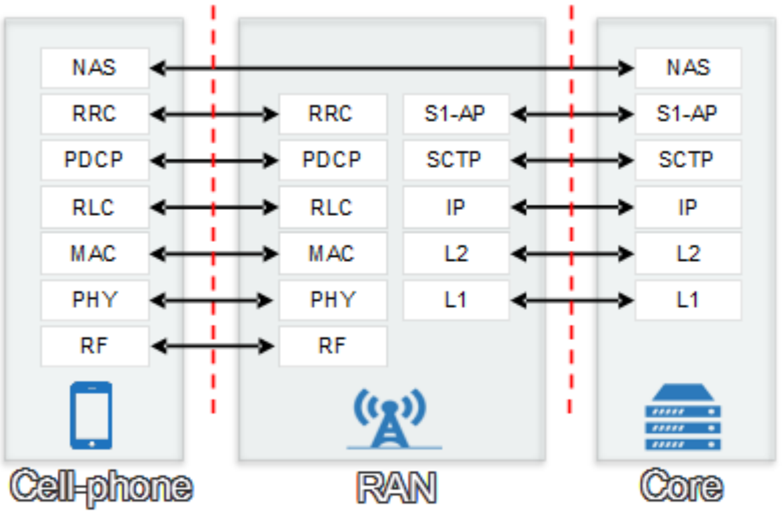


Figure 4 Traditional RAN

Figure 4 shows a traditional, hardware-based RAN system control plane (CP). The depicted layers are the following: Radio Frequency (RF), Physical (PHY), Media Access Control (MAC), Radio Link Control (RLC), Packet Data Convergence Protocol (PDCP), Radio Resource Control (RRC), Non-Access Stratum (NAS), Stream Control Transmission Protocol (SCTP), S1-Application Protocol (AP), IP and two data links protocols (L1 and L2). Part of the mentioned protocols can be virtualized and placed in different locations.

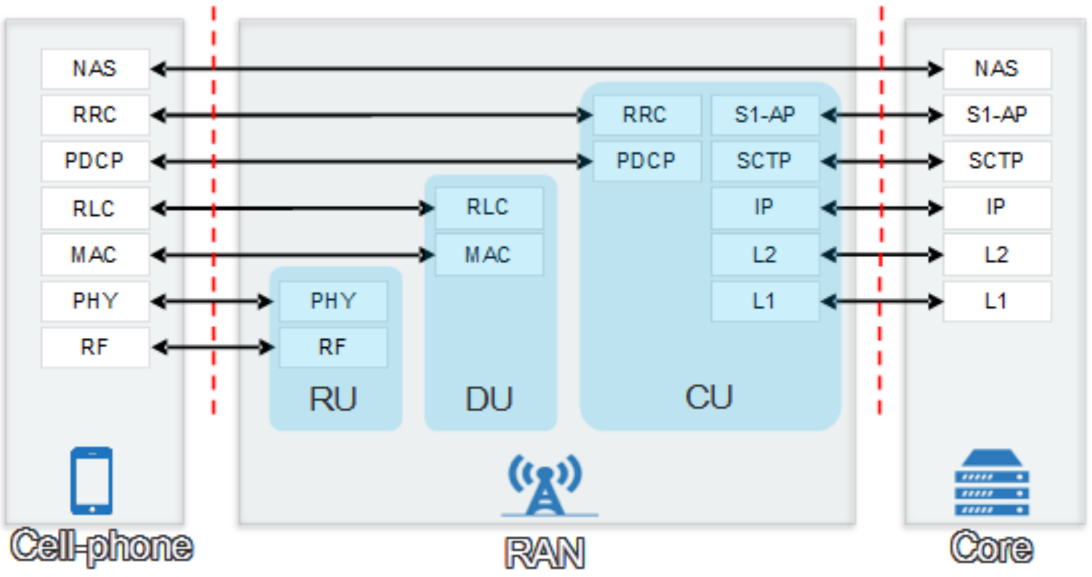


Figure 5 Cloud RAN Control Plane

Figure 5 shows the architecture of a Cloud RAN. The protocols mentioned above are included in three different *units*:

1. *Radio Unit* (*RU*), containing Physical (PHY-Lo) and Radio Frequency protocols
2. *Distributed Unit* (*DU*), containing Physical (PHY-Hi), RLC and MAC
3. *Central Unit* (*CU*), containing RRC, PDCP, S1-AP, SCTP, IP, L1 and L2.

It is important to notice that while DU and CU components are virtualized and can be placed in different locations (so called Virtualized Network Function - VNF), RU is tightly connected to the physical location (so called Physical Network Function - PNF).

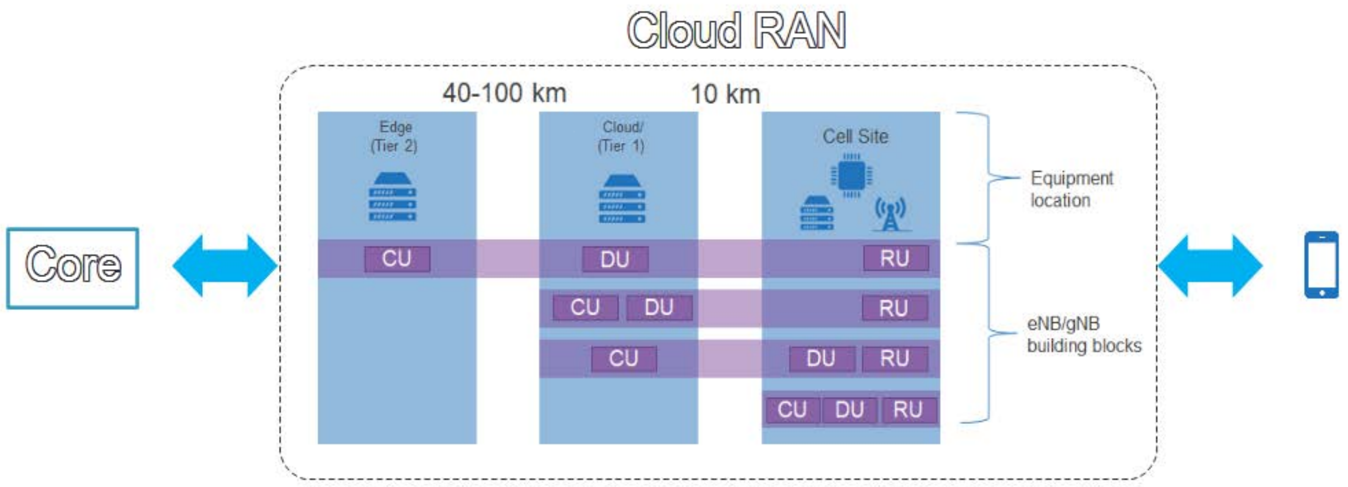


Figure 6 Cloud RAN Deployment

Figure 6 shows the deployment of an example of Cloud RAN. RUs cannot be virtualized since they include antennas, while the other two components, CU and DU, are virtualized: the approach is *hybrid*.
Figure 7 shows the details of the current deployment, without using MORPHEMIC: two infrastructures are used (OpenStack is much more used for the moment) and an orchestrator performs the mapping according to a network descriptor.

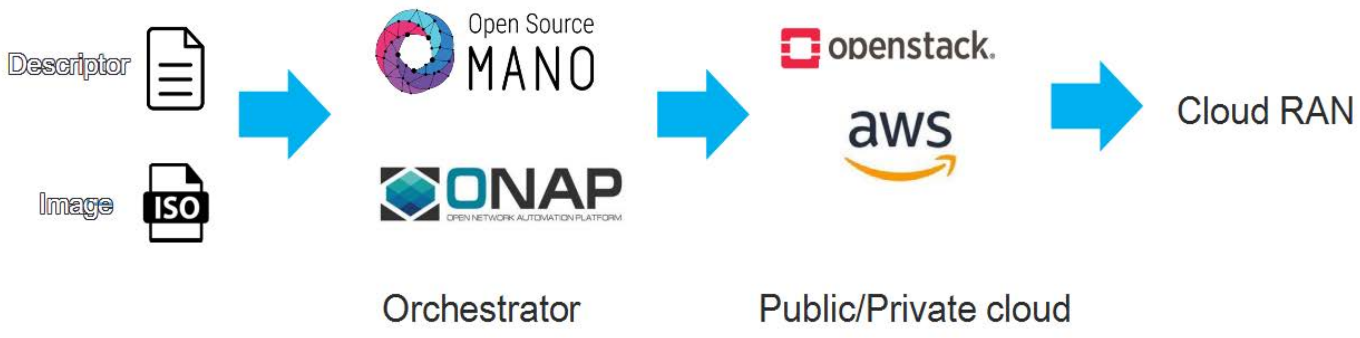


Figure 7 Cloud RAN Deployment details

The described *Software Defined RAN* is used to implement 5G base stations (*gNB*) deployable on demand on one or multiple sites thanks to the virtualization paradigm.
Two scenarios have been proposed:

1. Case A: the *static scenario* is focused on static deployment of the RAN; in other terms, according to the required infrastructural capabilities, MORPHEMIC will determine the best polymorphic application deployment configuration on edge and cloud (Figure 8)
2. Case B: the *dynamic scenario* is focused on the automatic modification of the deployment as the context changes. This scenario is the most complete one, since, along with the *polymorphic* features, it also exploits the feature of dynamic deployment modification and potentially *Proactive Adaptation*.

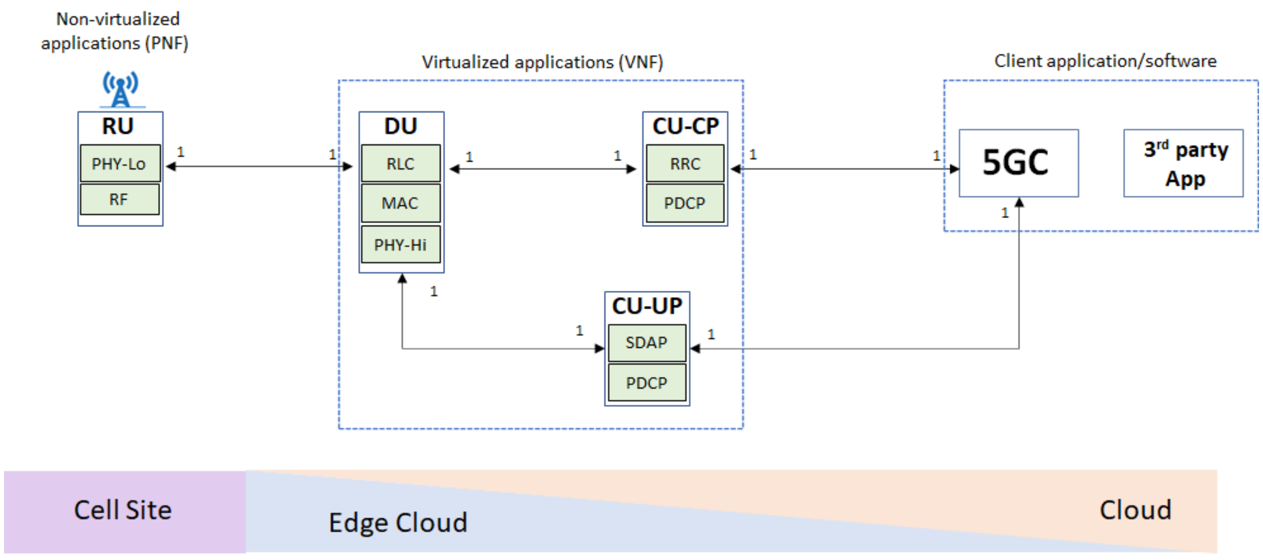

Non-virtualized applications (PNF)
Virtualized applications (VNF)
Client application/software
RU
PHY-Lo
RF
DU
RLC
MAC
PHY-Hi
CU-CP
RRC
PDCP
CU-UP
SDAP
PDCP
5GC
3rd party App
1
Cell Site
Edge Cloud
Cloud

Figure 8 Scenario of 5G cloud-RAN use-case

Figure 8 and Figure 9 show the deployment of the use-case. It must be noted that in this deployment CU component can be further divided into control plane (CP) and user plane (UP) parts - CU-CP and CU-UP respectively. They both provide CU functionality but can be deployed independently.

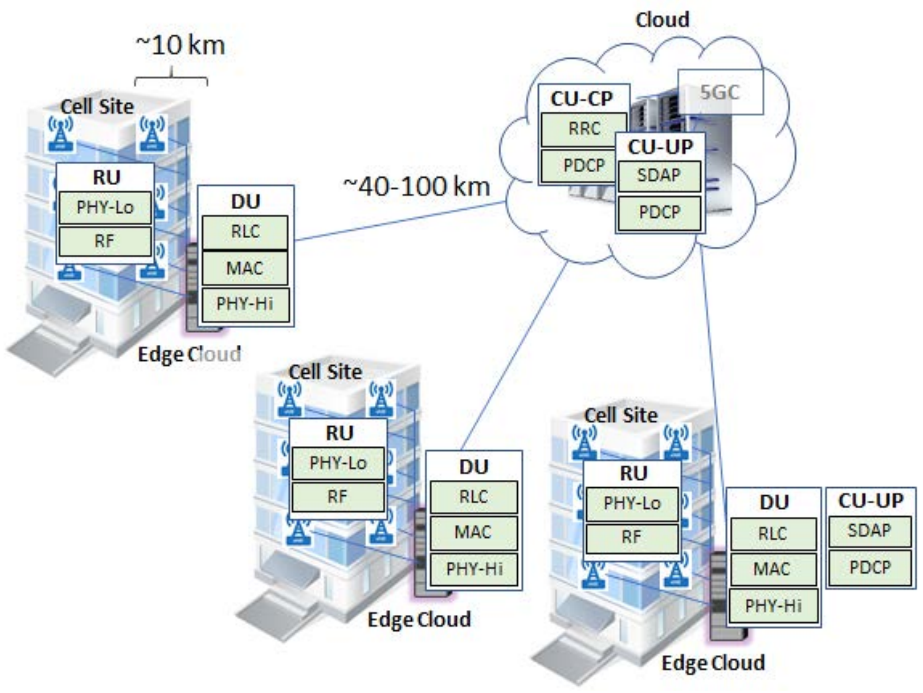


Figure 9 Cloud RAN: example of deployment

Both the pictures refer to three sites, specifically:

- The **Cell Site**, which hosts the antenna (RU) communicating with the end device over the air interface. A Cell Site can be 'thin' containing only RU or 'thick' gathering all the SD-RAN components (RU, DU, CU).

- The **Edge**, defined as "*all computing outside cloud happening at the edge of the network*"[10]. It can host DU and CU. It is usually located within 10 km from the Cell Site: this distance allows communication between DU and RU in less than 0.1 ms.
- The **Cloud,** located up to 100km from the Cell Site. It hosts CU (CU-CP) for which delays at the level of 10 ms are acceptable.

As mentioned, RU, which is the unit closest to the user, in general cannot be virtualized. Indeed, the VNF applications (DU and CU), which can be deployed at different levels from Cell side to Cloud, through Edge, can be virtualized. As shown in Figure 8, the core processing and, more in general, the 3rd party applications are at Cloud side.
It is important to notice that the RAN is not an *application by itself*, even if it can be seen as an application by MORPHEMIC: it should be considered as a Network as a Service (NaaS).

CU and DU are in *one-to-many* relationship: for this reason, the site holding CU unit must, in addition to enhanced processing capabilities, assure efficient large-scale traffic aggregation. The placement of the SD-RAN components is a trade-off between latency, performance and cost as well as depends on the actual scenario.

SD-RAN can be seen as part of an end-to-end slice of the 5G network. The deployment in Figure 9 is an example of this perspective. In this case the slice represents a private network where a 5G RAN is deployed in several buildings of a company located in a single geographical area. Cell Sites can be located in every room: it would consist of an RU unit with only one antenna (RF) and PHY-Lo. Therefore, in this case the RU would be lightweight. Tight relationship between PHY-Lo and RF would, however, render the RU a physical network function (PNF), i.e., a non-virtualized component. The RUs communicate with the DU unit located on an Edge Node within a building. The DU in that case would consist of PHY-Hi, MAC and RLC. It would be bound for low latency communication with the RUs. On the regional level the CU typically performs a high-level processing of the user and control flow coming from the underlying DUs. In this case the CU would consist of PDCP and RRC protocol layers in case of CU-CP and PDCP and SDAP in case of CU-UP. The same Cloud location could host 5G core (5GC) elements to enable end-to-end communication. The amount of computing power required for both CU and DU depends on the number of users using the 5G network. Both the CU and the DU could be perceived as VNFs.

3.1.2 Benefits introduced by MORPHEMIC

Currently, the deployment of Cloud RAN is performed as defined in the schema in Figure 6 and Figure 7. Specifically, the virtual infrastructures are managed through the orchestrator by using static descriptors.

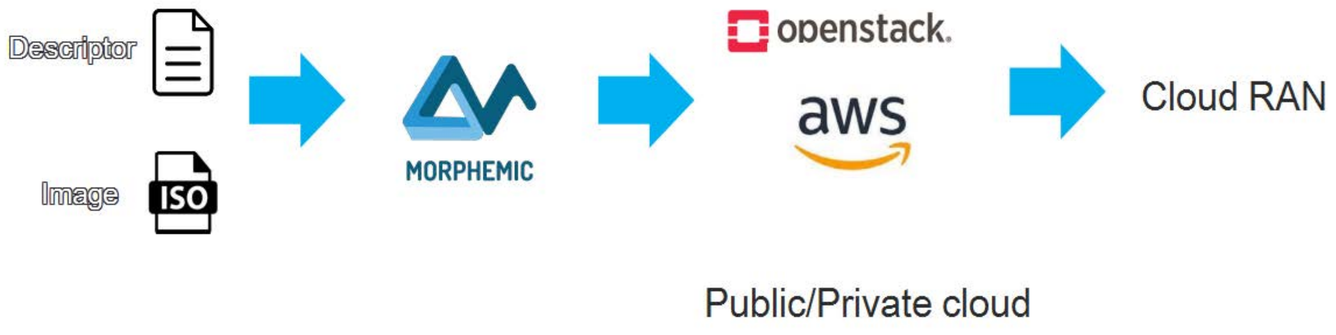


Figure 10 Cloud RAN deployment through MORPHEMIC

Figure 10 shows the deployment schema through MORPHEMIC: it will replace the orchestrator providing Polymorphic Adaptation and capability to dynamically modify the deployment configuration.
The benefits expected by the introduction of MORPHEMIC depend on the specific scenario. In particular, the *static scenario* (Case A) is focused on the optimization of a deployment that should *persist* for a certain amount of time. For this case, the static optimization of the deployment on heterogeneous environments (**Polymorphic Adaptation**), the performance level (**Hardware Accelerator Support**) and a usable UI (**Uniform User Interface**) are considered

[10] Karim Arabi

added values. For the *dynamic scenario* (Case B) the aforementioned benefits are still important, since the core features of the use-case are the same; however, also the more dynamic benefits, i.e., **Proactive Adaptation** and **Self-Healing Capability** will be important to support the quickly changing context.

Table 5 summarizes the details.

Table 5 Benefits introduced by MORPHEMIC

Feature	Case A (static)	Case B (dynamic)
Polymorphic Adaptation	✓	✓
Proactive Adaptation		✓
Self-Healing Capability		✓
Hardware Accelerator Support	✓	✓
Uniform User Interface	✓	✓

Both the cases are very complete, but the Case B covers all the features of MORPHEMIC. This suggests to use Case B as the basis for the analysis of the use-case requirements.

3.1.3 Requirements

MORPHEMIC will have to support the deployment of an SD-RAN system as described in the previous sections. The features of such a system suggest some important characteristics of MORPHEMIC. Specifically, the support for virtual networks requires capabilities to take care of latency, hardware and dynamicity in the deployment.
In particular, since the SD-RAN is constituted by a set of virtualised and real components, the first aspect to be supported by MORPHEMIC is *a hybrid deployment configuration*, enabling to deploy applications on cloud and on hardware components. Hardware components are needed for some physical constraints (i.e., antennas), but also to reduce latency. However, it is possible to keep low latency and assure a certain processing power also by using virtual components and MORPHEMIC must be able to determine and enforce deployment configurations considering this aspect.
In summary, to support 5G RAN use-case, MORPHEMIC will have to:

- *support* hardware-based elements, such as PNF, as application deployment environment
- keep control on the latency at *Edge* level
- support FPGA acceleration at *Cloud* level.

SD-RAN slice is designed to represent some connectivity capability serving a particular application. Therefore, deployment of SD-RAN slice would be often coupled with the deployment of an application. A service or end-user application would reside on the end device side (i.e., a mobile phone or a machine) as well as network side (i.e., collocated with RAN components - in that case we would consider the MEC deployment) or remote cloud. Part of the required work can be done by MORPHEMIC that will be able to efficiently deploy part of SD-RAN.
Table 6 shows a list of requirements obtained from the analysis of the description of the use-case.

Table 6 MORPHEMIC requirements related to SD-RAN use-case

ID	Name	Description	Topic	Priority (MoSCoW)
UC-C-SE.1	Support for non-virtualized components	MORPHEMIC must support hardware components, such as Cell Sites	Supported Environments	MUST
UC-C-SE.2	Multi-site deployment	MORPHEMIC should support deployment configurations spanning	Supported Environments	SHOULD

		across multiple geographical locations		
UC-C-UF.1	Targeted deployment: network capability	MORPHEMIC could determine and enforce a deployment configuration according to the requirements of network capability	Parameters for the Utility Function	COULD
UC-C-UF.2	Targeted deployment: price	MORPHEMIC should determine and enforce a deployment configuration according to the requirements of price	Parameters for the Utility Function	SHOULD
UC-C-UF.3	Targeted deployment: packaging	MORPHEMIC should determine and enforce a deployment configuration according to the requirements of packaging (i.e., container)	Parameters for the Utility Function	SHOULD
UC-C-UF.4	Geographical awareness	MORPHEMIC could support geographical constraints in the deployment configuration	Parameters for the Utility Function	COULD
UC-C-UF.5	Targeted deployment: computing power	MORPHEMIC must determine and enforce a deployment configuration according to the requirements of computing power	Parameters for the Utility Function	MUST
UC-C-SEC.1	Support for traffic isolation	MORPHEMIC could determine and enforce deployment configurations supporting traffic isolation	Security	COULD
UC-C-SEC.2	Support for secure communications	MORPHEMIC could determine and enforce deployment configurations supporting secure communications among deployed components (i.e., private networks)	Security	COULD
UC-C-SEC.3	Support for security related applications	MORPHEMIC could determine and enforce deployment configurations supporting security elements (such as firewalls, IDS etc)	Security	COULD
UC-1-SE.1	Platform awareness: DPDK	MORPHEMIC should support DPDK solutions	Supported Environments	SHOULD
UC-1-SE.2	Platform awareness: CPU pinning	MORPHEMIC should support CPU pinning solutions	Supported Environments	SHOULD
UC-1-SE.3	Support for redundancy	MORPHEMIC could be able to determine and enforce	Supported Environments	COULD

		a redundant deployment configuration based on the use-case requirements concerning reliability		
UC-1-SE.4	Support for Fault Management	MORPHEMIC should be able to deploy Fault Management tools	Supported Environments	SHOULD
UC-1-SE.5	Support for Configuration Management	MORPHEMIC should be able to deploy Configuration Management tools	Supported Environments	SHOULD
UC-1-SE.6	Support for Accounting Management	MORPHEMIC should be able to deploy Accounting Management tools	Supported Environments	SHOULD
UC-1-SE.7	Support for Performance Management	MORPHEMIC should be able to deploy Performance Management tools	Supported Environments	SHOULD
UC-1-SE.8	Support for Security Management	MORPHEMIC could be able to deploy Security Management tools	Supported Environments	COULD
UC-1-UF.1	Support for low latency in terms of deployment time	MORPHEMIC could determine and enforce production-ready deployment configurations with a minimal deployment latency	Parameters for the Utility Function	COULD
UC-1-UF.2	Targeted deployment: latency between the deployed components	MORPHEMIC could determine and enforce a deployment configuration according to the requirements of the latency in communication between the components	Parameters for the Utility Function	COULD
UC-1-AD.1	Dynamic deployment configuration	MORPHEMIC must dynamically re-determine the deployment configuration to react to changing situations (i.e., scaling up/down, migrating)	Adaptation	MUST
UC-1-AD.2	Live migration	MORPHEMIC could apply the new determined deployment configuration minimizing the downtime of the service	Adaptation	COULD

3.1.3.1 Requirements group: deployment environment type

SD-RAN use-case is founded on applications which need to be deployed on different environments, not only virtual. As mentioned in section 3.1.1, some elements, such as Cell Sites, including antennas, cannot be deployed on virtual environments. The requirements UC-C-SE.1, UC-C-UF.1 and UC-C-UF.3 describe the deployment environments for this use-case. Specifically, the possibility to take into account the network capability would be an added value, such as the redundancy in order to increase the availability.

3.1.3.2 *Requirements group: deployment environment's features*

This use-case suggests to consider price, security, latency and traffic for the deployment and the Utility Function (section 2.3). Specifically, latency, computing power and traffic (UC-1-UF.2, UC-1-SE.1, UC-1-SE.2 and UC-1-UF.1) are parameters that can vary quickly and, even if the traffic increases, latency should be kept under a reasonable threshold. Computing power (UC-C-UF.5) must be taken into account to keep the costs under control, price (UC-C-UF.2) is also important, since it is a business goal. Security (UC-C-SEC.1, UC-C-SEC.2 and UC-C-SEC.3) is not directly inside the scope of MORPHEMIC, but the possibility to include in the deployment tools that guarantee data, network security, focused and traffic isolation will be an added value.

3.1.3.3 *Requirements group: geographical awareness*

Network related functionalities are affected by the geographical location of the deployed elements. In this sense, requirements UC-C-UF.4 and UC-C-SE.2 guarantee that the deployment takes into account the geographical constraints. This can be connected with the latency, but also with legal constraints, such as GDPR that restricts the geographical area of storage affecting also the transmission.

3.1.3.4 *Requirements group: dynamic management and configuration*

This group of requirements (UC-1-AD.1, UC-1-AD .2, UC-1-SE.3 and UC-1-SE.4-8) extends the original idea of MORPHEMIC. The capabilities to detect the potential issues of the environment in terms of security, performance and fault and to modify the deployment configuration according are among the best potential added values.

3.1.4 Technological requirements

Components of 5G Cloud RAN are currently up and running, manually deployed and dimensioned in static way according to the experience of IS-W and to the requirements of the customers. As for the other use-cases, the business objective is clear and the parameters, including the constraints, defining the *Utility* (section 2.3) are clear.

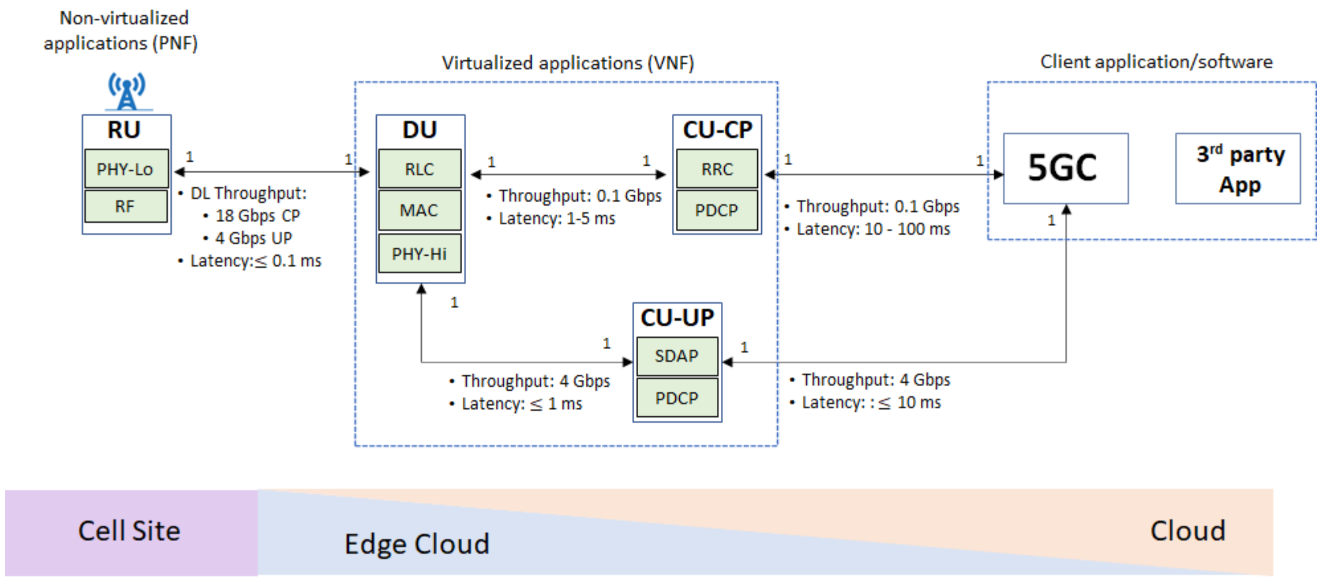


Figure 11 Static scenario of 5G cloud-RAN use-case with requirements

Figure 11 is identical to Figure 8 but includes also the requirements in terms of throughput and network latency. They are part of the constraint on which the Utility Function will be built. Besides this, in most of the scenarios experienced, a possible static deployment configuration presents the following features:

- The DU and the CU (considering as CU-CP and CU-UP) as a scalable number of VM, each of which would require up to **2GB of RAM** and **30-60 threads**

- Number of CPU cores strictly dependent on the requirements of the customer, but support for HW accelerator
- Max CP throughput **18 Gbps**, Max UP throughput **4Gbps**
- The most stringent latency constraint is **0.1 ms** in the communication between RU and DU.

Part of the listed *numbers* have been used in Table 6 to better define some *constraints* helpful for the deployment. However, they are the basis to evaluate the *Utility Function* for each deployment configuration in order to be constantly as close as possible to the business goals.

3.2 E-Brain Science

This use-case, proposed by Centre Hospitalier Universitaire Vadois[11], is focused on the use of neuroimaging tools in the diagnosis and treatment of individuals suffering from brain injuries and disorders.
Specifically, objective of the case is to analyse data on populations of patients for diagnosis and research. Currently part of this task is performed through software solutions that clinicians, neuroscientists and epidemiologists can run on their laptop or desktop or, in case of large patient cohorts, on HPC. However, the increased complexity and data available will make necessary the use of cloud computing and GPUs to achieve the desired results in a reasonable amount of time. For this reason, MORPHEMIC will be used to make more efficient the use of computing and storage resources to obtain the same results keeping a high level of security.

3.2.1 Use-case Description

E-Brain Science's architecture is conceived to advance the fundamental and digital knowledge on healthy brain aging and neurocognitive disorders. Specifically, the provided functionalities will be:

1. establish a framework for **federating clinical data** within and across **data sources** (*hospitals*, *clinics* and *cohorts*)
2. develop **benchmarking technology** that respects **anonymity** requirements
3. evaluate **AI based diagnostic**
4. derive **biological signatures** of **brain diseases**.

More in general, the use-case concerns the analysis of a big amount of data coming from different data sources. As well as any healthcare data, these data are often *sensitive data* with stringent requirements in terms of security and anonymization. MORPHEMIC will not manage these aspects, but it could be possible to deploy an application covering these requirements.

[11] https://www.chuv.ch

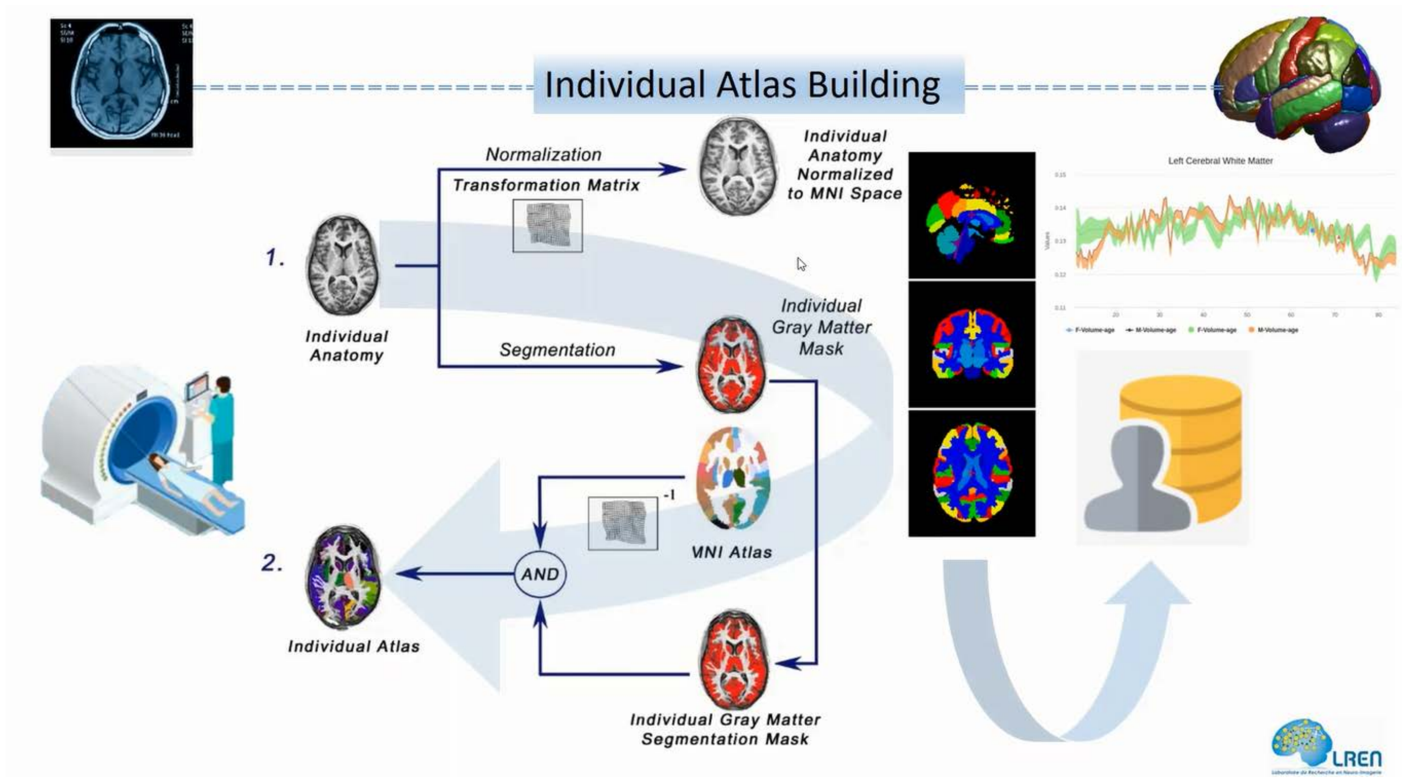

Individual Atlas Building
Normalization
Transformation Matrix
Individual Anatomy Normalized to MNI Space
1.
Individual Anatomy
Segmentation
Individual Gray Matter Mask
Left Cerebral White Matter
-1
MNI Atlas
2.
AND
Individual Atlas
Individual Gray Matter Segmentation Mask
LREN

Figure 12 E-Brain tools and features

Figure 12 shows the produced data and the tools used to retrieve them: those data should be processed and stored in an efficient and secure way.

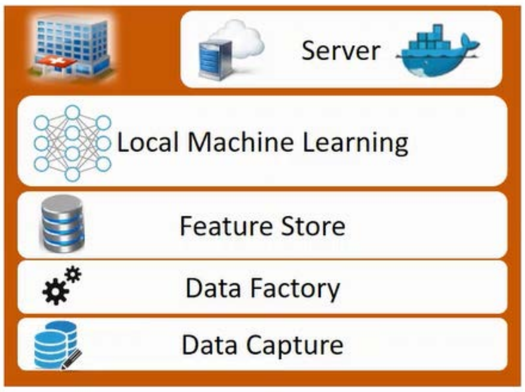


Figure 13 E-Brain Science use-case operations

Figure 13 shows the main functionalities that E-Brain Science use-case will provide. Data should be captured, processed, stored and used by applying machine learning algorithms. Currently these functionalities are available as a set of tools developed, deployed and provided to clinicians which will need to use them. The objective is to modify the deployment configuration to obtain a network of interconnected local instances, as shown in Figure 14.

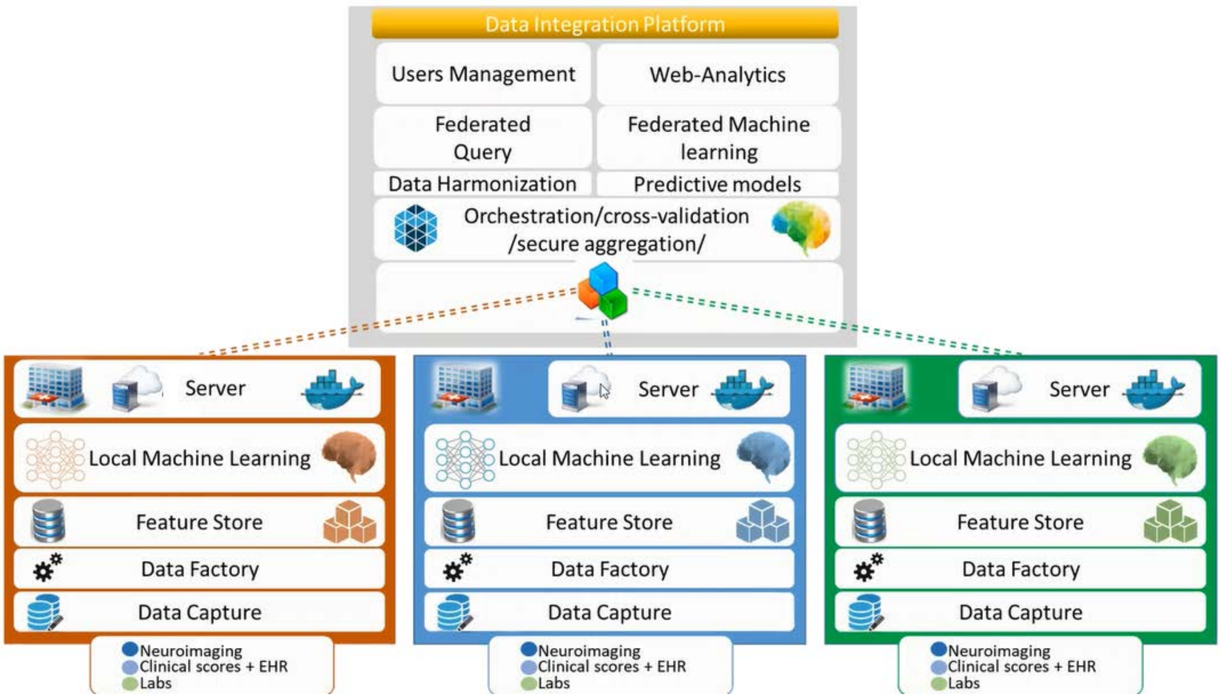


Figure 14 Distributed instances of E-Brain tools

The local instances are interconnected through a *data integration platform* providing users with access and processing capabilities to run the *local machine learning* instances and aggregate the obtained results. This deployment allows to cross process and compare data coming from different data sources (hospitals) increasing the reliability of the results. Figure 15 shows the architecture of the use-case. The services on the upper side represent three sub-cases:

- **image pre-processing pipeline**, which consists in providing standardized workflow for pre-processing neuroimaging data. The users will be able to select and configure neuroimaging workflows from data conversion, in order to segment the images and extract the brain features
- **SPM on web** provides a web tool for sharing and visualization of image analysis conducted with SPM (Statistical Parametric Mapping), the most popular open-source package for neuroimaging analysis
- **federated machine learning** consists in providing an innovative system that wide users (clinicians, neuroscientists, epidemiologists) can access and use to analyse clinical and research data without moving them from the hospital or private cloud servers where they reside and without infringing on patient privacy. The strategy is to use cloud computing and machine learning approaches and create a meeting place for neuroscience and IT for collaborative brain disease research as well as benefitting clinicians on a daily basis.

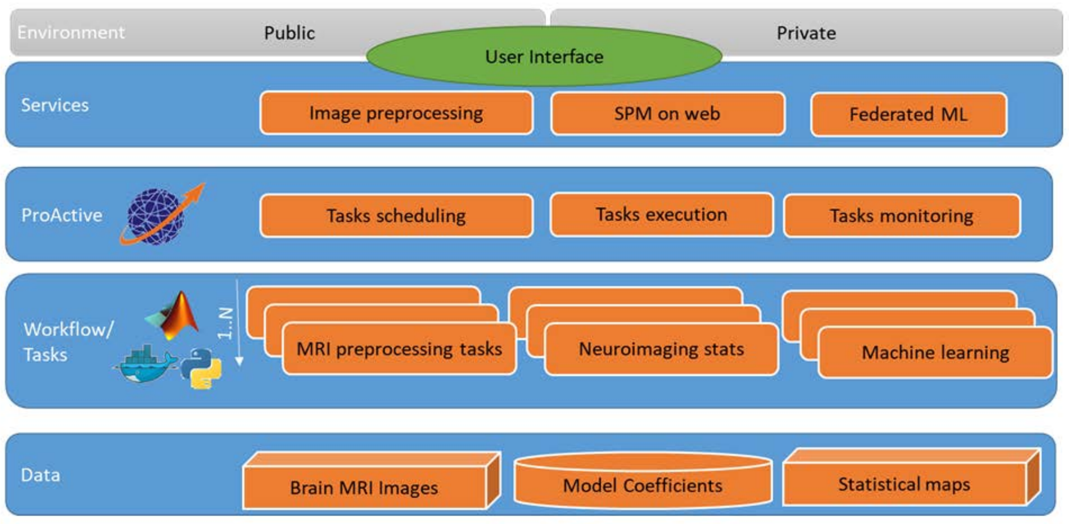


Figure 15 Architecture of E-Brain use-case

The environment can be public or private, basing on the requirements of the data in terms of storage location and privacy. The task management modules are implemented by **ProActive** and consist in scheduling, execution and monitoring. These tasks are implemented by workflows including MRI pre-processing, neuroimaging stats and machine learning. Finally, data can be private (such as brain MRI images) or shared (such as model coefficients and statistical maps) according to privacy regulations.

The first sub-use-case is focused on *image pre-processing pipeline*.

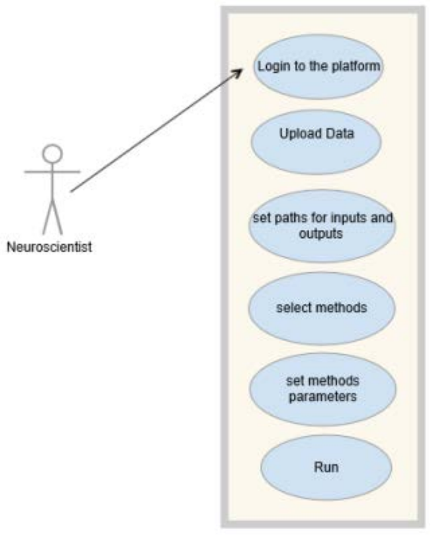


Figure 16 E-Brain sub-use-case 1 (image preprocessing)

Figure 16 shows the steps. In particular the images are uploaded after the successfully log into the platform, processed and the results are downloaded. The main criteria for this use-case is the basic paradigm of cloud computing, i.e., *if you have more data you need more computational resources*. The pre-processing operations can take different amount of time, from a few minutes to several hours.

The second sub-use-case, SPM on the web, is more interactive. Figure 17 shows the details: in general, the user, after completed the SPM analysis locally, uploads the output on the web application to shares it. The results can be visualized by other users which collaborates on the same topic, according to some visualization settings.

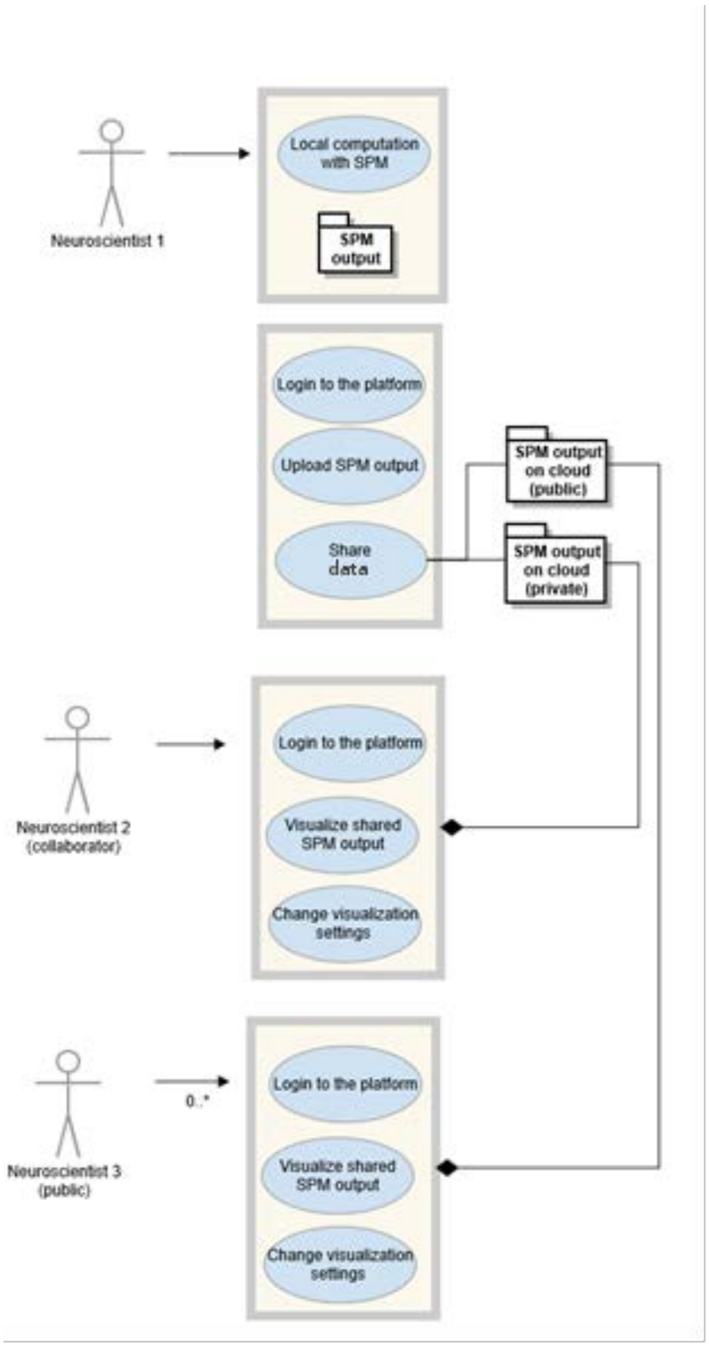

Neuroscientist 1
Local computation with SPM
SPM output
Login to the platform
Upload SPM output
Share data
SPM output on cloud (public)
SPM output on cloud (private)
Neuroscientist 2 (collaborator)
Login to the platform
Visualize shared SPM output
Change visualization settings
Neuroscientist 3 (public)
0..*
Login to the platform
Visualize shared SPM output
Change visualization settings

Figure 17 E-Brain sub-use-case 2 (SPM on the web)

Figure 18 shows the third sub-use-case, *Federated Machine Learning*. This sub-use-case is focused on data locally stored, at hospital side, along with public data, at cloud side, which are analysed by using ML algorithms. Specifically, some dataset can be used for training and other for model validation, obtaining reports and results.

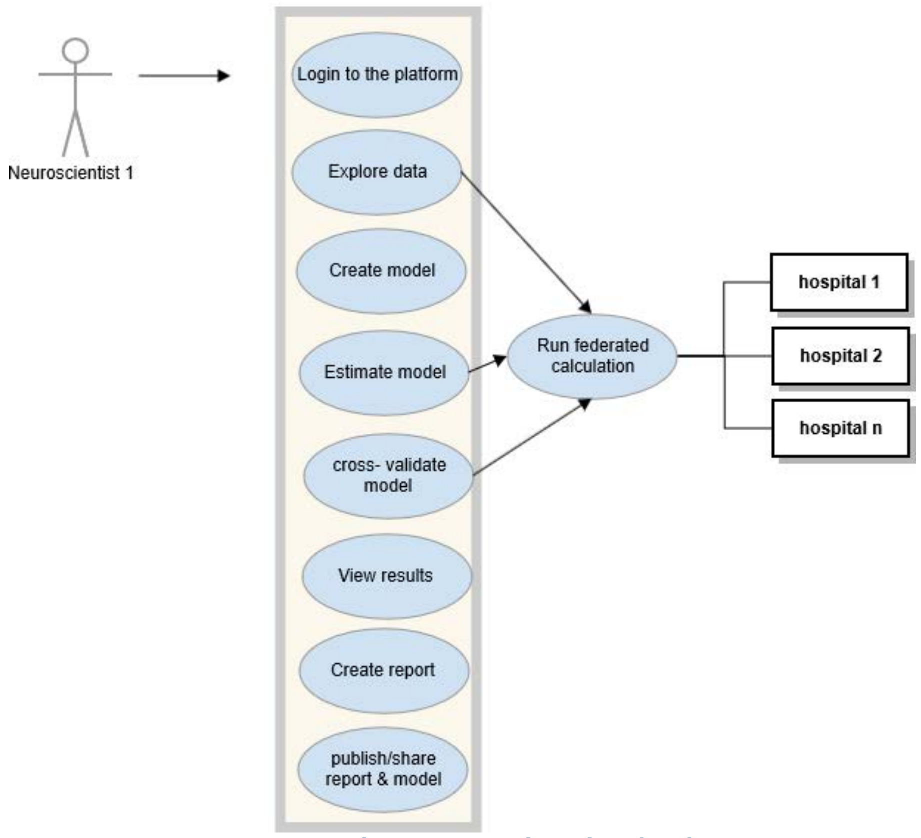


Figure 18 E-Brain sub-use-case 3, Federated machine learning

3.2.2 Benefits introduced by MORPHEMIC

The main challenges of this use-case regard the usage of data and their privacy level. Specifically, a trade-off between capabilities provided by the Cloud and data privacy is needed.

Neuroimaging need to be processed by using specific algorithms that should be validated against established results and codes and tuned for optimal results. The specific requirements, especially in terms of security, of the clinical data suggest to run the algorithms as locally as possible. However, the high processing capability required forces the clinicians with large patient cohorts to use HPC: sometimes, the increased complexity and data available makes necessary the use of GPUs to achieve the desired results in a reasonable amount of time. In this sense ProActive provided a cloud-based task management tool, in order to move part of the operations (and data) on the Cloud: MORPHEMIC will give a further added value through the capability to deploy software-as-a-service in community (public cloud) and private (private cloud) execution environment. This possibility will combine a potentially unlimited data processing capabilities with the desired privacy level.

Specifically, in order to correctly deploy E-Brain Science use-case, MORPHEMIC will have to provide:

- *support for centralized access*: since the use-case guarantees a single access point for software and data deployed in private execution environments
- *support for private environments*: since the use-case guarantees certain tasks, including data pre-processing, brain feature extraction to data mining and storage, will be performed
- *support for community execution environment*: since advanced ML algorithms and predictive analysis need high capabilities and optimization

- *support for the deployment of master orchestrator components* to be run in community execution environment: the use-case needs to fetch the aggregate results of the algorithms executed in the private execution environments and aggregate them in a cross-centre data analysis result.

Data security is very important in healthcare context. Healthcare data are classified as *sensitive personal data* and, according to GDPR, must comply to very restrictive rules, in term of anonymisation and storage location.
Part of these aspects can be managed by MORPHEMIC by defining some constraints on data storage location. More in general, the capability to include in a deployment, features or tools to guarantee security and privacy could significantly improve the value of the MORPHEMIC platform. Some nice-to-have requirements described in section 3.2.3 regard such a capability.
The three listed sub-use-cases need proactive, multi-environment support (**Polymorphic** and **Proactive Adaptation**). The sub-use-cases 1 and 3 could require high processing power, so **Hardware Accelerator Support** could be requested. Self-Healing Capability and Uniform UI depend on specific features of the sub-use-case in terms of dynamicity and interaction.
Table 7Table 7 Benefits introduced by MORPHEMIC for E-Brain use-case summarizes the benefits introduced by vMORPHEMIC for E-Brain use-case.

Table 7 Benefits introduced by MORPHEMIC for E-Brain use-case

Feature	Sub-use-case 1	Sub-use-case 2	Sub-use-case 3
Polymorphic Adaptation	✓	✓	✓
Proactive Adaptation	✓	✓	✓
Self-Healing Capability	✓		
Hardware Accelerator Support	✓		✓
Uniform User Interface		✓	✓

3.2.3 Requirements

E-Brain Science use-case deals with healthcare data: the two main requirements are, in this sense, availability of high processing power and security. The processing power is needed because neuroimaging data processing can include jobs of several hours that need specific capabilities, including hardware acceleration and GPU. Security, due to sensitive personal data rules, can impact on data processing and storage location along with encryption and secure access.
Some of the requirements provided on Table 8 are the same of Table 6: they will be identified with the same ID.

Table 8 MORPHEMIC requirements related to E-Brain use-case

ID	Name	Description	Topic	Priority (MoSCoW)
UC-C-SE.1	Support for non-virtualized components	MORPHEMIC must support hardware components, such as Cell Sites	Supported Environments	MUST
UC-C-SE.2	Multi-site deployment	MORPHEMIC must support deployment configurations spanning across multiple geographical locations	Supported Environments	MUST
UC-C-SE.3	Support for GPU	MORPHEMIC could support software or hardware solutions using GPU	Supported Environments	COULD
UC-C-UF.1	Targeted deployment: network capability	MORPHEMIC could determine and enforce a deployment configuration	Parameters for the Utility Function	COULD

		according to the requirements of network capability		
UC-C-UF.2	Targeted deployment: price	MORPHEMIC should determine and enforce a deployment configuration according to the requirements of price	Parameters for the Utility Function	SHOULD
UC-C-UF.3	Targeted deployment: packaging	MORPHEMIC should determine and enforce a deployment configuration according to the requirements of packaging (i.e., container)	Parameters for the Utility Function	SHOULD
UC-C-UF.4	Geographical awareness	MORPHEMIC must support geographical constraints in the deployment configuration	Parameters for the Utility Function	MUST
UC-C-SEC.1	Support for traffic isolation	MORPHEMIC could determine and enforce deployment configurations supporting traffic isolation	Security	COULD
UC-C-SEC.2	Support for secure communications	MORPHEMIC could determine and enforce deployment configurations supporting secure communications among deployed components (i.e., private networks)	Security	COULD
UC-C-SEC.3	Support for security related applications	MORPHEMIC could determine and enforce deployment configurations supporting security elements (such as firewalls, IDS etc)	Security	COULD
UC-2-SE.1	Support for ProActive	MORPHEMIC must be able to deploy ProActive	Supported Environments	MUST
UC-2-SE.2	Support for private execution environment	MORPHEMIC must be able to deploy on private execution environments (including private cloud)	Supported Environments	MUST
UC-2-SE.3	Support for Master orchestrator components on public environment	MORPHEMIC must be able to deploy master orchestrator components on public execution environment	Supported Environments	MUST

3.2.3.1 Requirements group: deployment environment type

E-Brain Science use-case is focused on data: in this sense the possibility to define the type of environment (virtual or non-virtual) gives the control on the location on which data are stored. In this sense, the requirements UC-C-SE.1, UC-C-UF.1, UC-C-UF.3 and UC-2-SE.2 describe the environments that required for this use-case. Particular focus is given on private execution environment, strictly linked with privacy requirements.

3.2.3.2 *Requirements group: deployment environment's features*

ProActive, by Activeeon, is the technological solution on which E-Brain use-case is founded. Requirement UC-2-SE.1 explicitly asks for this support. Very similar, but more generic, is requirement UC-2-SE.3, which focuses on *master orchestrator components* that can integrate ProActive to support E-Brain. Finally, the possibility to support environments with specific hardware features (UC-C-SE.3) is considered a *nice-to-have* feature that MORPHEMIC could provide to better support other use-cases. The aspects related to security (UC-C-SEC.10, UC-C-SEC.2 and UC-C-SEC.3), as said before, are not directly supported by MORPHEMIC, but it could be possible to deploy security-related tools.

3.2.3.3 *Requirements group: geographical awareness*

The General Data Protection Regulation (GDPR) is a critical feature for applications belonging to healthcare domain: sensitive personal data must be treated with special care and the full control of geographical location is one of the explicit requests to be compliant with the regulation. In this sense, the requirements UC-C-UF.4 must be satisfied, while UC-C-SE.2 is an important added value.

3.2.4 Technological requirements

Concerning the operations to be performed on the medical data produced by a single subject, the current version of E-Brain science is deployed on:

- **Hardware**
  - o Processor 48x Intel(R) Xeon(R) CPU E5-2680 v3 @2.50 GHz
  - o Memory 136 GB

- **OS**
  - o Kernel Linux 4.4.0.142-generic (x86 64-bit)
  - o Distribution Ubuntu 16.0406 LTS

Concerning data, Table 9 shows the component processes, the data flow amount of input and output data and the running time. Each process should be seen as a non-mandatory step of a pipeline.

Table 9 Data and processes of E-Brain Science use-case

Processes of E-Brain Science use-case	in Data	out Data	Running time
DCM2NII (common input for all other steps)	2.5 GB	764 MB	12 min
MPM	484 MB	606 MB	21 min
NeuroM	44 MB	2.9 MB	12 min
NeuroMproj	205 MB	7.3 MB	32 sec
fMRI	140 MB	592 MB	8 min
DWI_preproc	212 MB	187 MB	42 min
DWI_calc	260 MB	156 MB	30 min
+ 9 prepocessing steps		~kB	~2.5 min

3.3 Computational Fluid Dynamics simulation

This use-case, proposed by ICON[12], regards Computational Fluid Dynamic Simulation. Specifically, MORPHEMIC will support the deployment of three CFD scenarios different in terms of requested resources.

3.3.1 Use-case Description

ICON's Computational Fluid Dynamics (CFD) tools are used to compute flow quantities in a wide range of industrial applications, like aerodynamics in the automotive and aerospace industry, free surface flows, species transport, multi

[12] https://www.iconcfd.com

component flows as well as thermal simulations. These kinds of simulations are still demanding in spite of hardware usage and simulation runtimes. There will be defined three different setup scenarios Small, Medium and Large (S/M/L) for the use case, described in detail in the upcoming deliverable D6.3.

- *Large* - high-fidelity simulations are very demanding in terms of resources, including memory and network bandwidth and typically require an HPC cluster (128 – 2000+) with high-speed interconnect.
- *Medium* - medium-fidelity simulations can be run on high core count (16-128) shared-memory machines or on HPC clusters.
- *Small* - low-fidelity simulations can be run on single workers with few compute cores (<16).

ICON aims to ensure that the users of their web-based framework, iconCFD Platform, get their results within a timeframe agreed in their SLA (for example 24 hours) and at a minimum cost. On this sense, MORPHEMIC should be able to pro-actively adapt the number and type of workers to make sure that each job will be completed within the specified time, for each scenario. The simulations can be run on x86-64 (AMD, INTEL), or 64-bit Arm architectures of varying number of cores, or on clusters of these. The best data storage strategy needs to be evaluated, if it can be for instance local NFS storage, AWS S3 type data lake storage, MORPHEMIC will mount and manage storage and/or hybrid storage with local cache in VMs. This depends on the data volume, the access frequency of the different types of simulations.

In this use-case, the MORPHEMIC platform performs the deployment of appropriated workers (VMs, cloudHPC) for the different scenarios on demand. The application will contain a run-manager (scheduler) which will use a "FIFO" approach with three queues, one for each scenario. The iconCFD Platform frontend is responsible for the case creation and its submission via the scheduler/run-manager. The software for the workers will be packaged in a Docker or singularity container: Figure 19 shows a schematic view of the use case scenario.

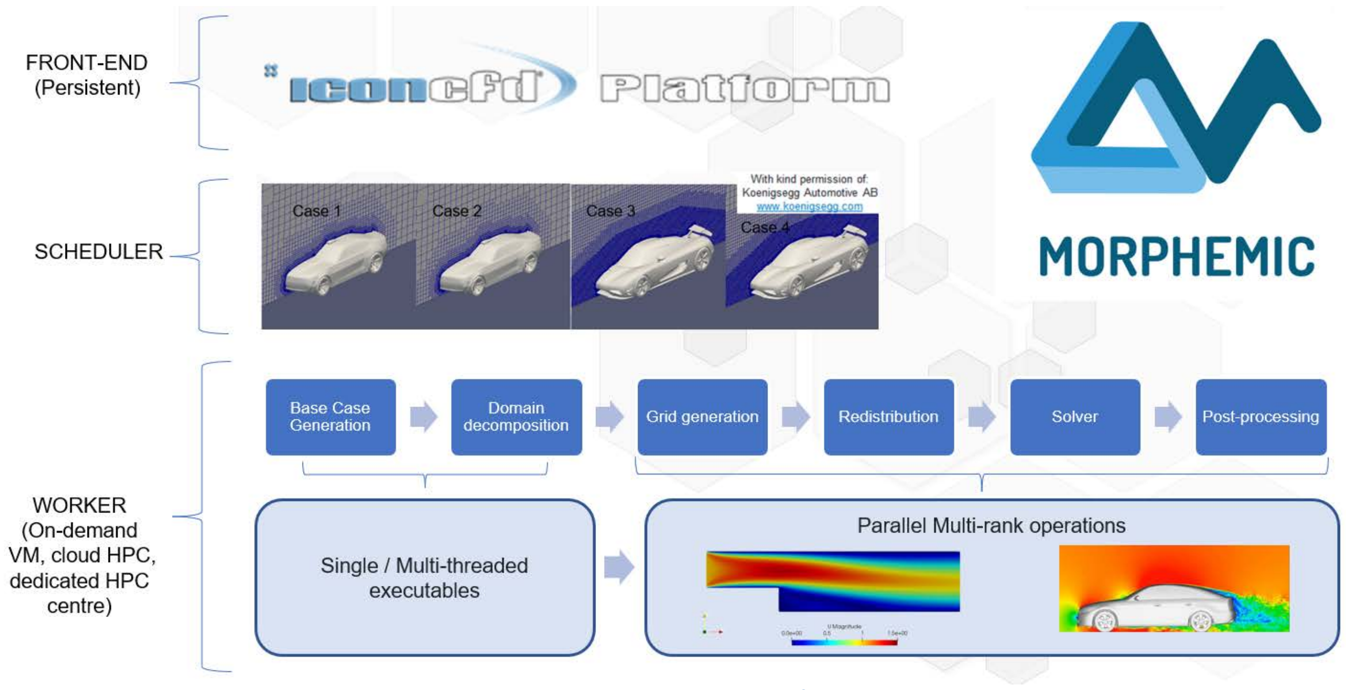

FRONT-END (Persistent)
iconCFD Platform
MORPHEMIC
SCHEDULER
With kind permission of: Koenigsegg Automotive AB www.koenigsegg.com
Case 1
Case 2
Case 3
Case 4
WORKER (On-demand VM, cloud HPC, dedicated HPC centre)
Base Case Generation
Domain decomposition
Grid generation
Redistribution
Solver
Post-processing
Single / Multi-threaded executables
Parallel Multi-rank operations

Figure 19 CFD use-case workflow

3.3.2 Benefits introduced by MORPHEMIC

MORPHEMIC will be used to deploy the application managing a queue of Small, Medium and Large simulations.

The benefits of the project for ICON are:

- Easy deployment
- No further custom setup for each cloud provider/machine type etc...
- Reduction of cost by using best offer
- Reduction of cost by reducing administration and maintenance efforts
- Ensure simulations are done in a given time frame.
- Simple application specification/definition with the CAMEL.

In general, a simulation, that is a high consuming application, can take long depending on the context and on its features. In other terms, the duration of a simulation may depend on the required cost or on its complexity or on both. For some simulations there are SLAs in place that limit the maximum acceptable duration. These SLAs usually depend also on costs: the higher are the costs, the higher are the expected SLAs.

Polymorphic feature and pro-Active adaptation will be very useful for our applications that can be created on-demand and deployed according with the requirements described and to minimize the costs whilst adhering to the SLAs.

Self-Healing Capability is very important at least for Large case simulations, since it is critical to find potential issues and adapt the deployment according. It is still under analysis if Hardware Accelerator Support could be useful for the iconCFD solvers. In that case, probably the biggest benefit will be given to Medium and Large simulations, more expensive in terms of requested computing resources.

The MORPHEMIC's technology could be adapted in future to bring in machine learning algorithm to predict specific flow quantities based on the training data gathered with the CFD computations.

Table 10 shows, for each application, the functionalities provided by MORPHEMIC that will give benefits to iconCFD platform.

Table 10 Benefits introduced by MORPHEMIC for CFD use-case

Feature	Small/Medium/Large
Polymorphic Adaptation	✓
Proactive Adaptation	✓
Self-Healing Capability	✓
Hardware Accelerator Support	(✓)
Uniform User Interface	

3.3.3 Requirements

ICON's requirements for MORPHEMIC is to have the possibility to specify constraints and cost functions to fulfil the requirement to dynamically submit simulations for a given queue in less than the SLA time at minimum cost. The hardware requirements for the workers are for the Medium and Large simulation scenario either a shared memory cluster or a HPC cluster with shared file system and high speed inter connect network. MORPHEMIC needs to provide a mechanism to set dynamically the minimum size of workers needed.
Small applications are quicker simulations whose objective is to obtain the best possible result without taking much time. For the ICON use case the key is to develop a scheduler/run manger which organize the handling of the different type workers and their data exchange with the iconCFD Platform frontend. This scheduler will be always up and running, whereas the number of workers change during the runtime of the deployed application.

According to the requirements, MORPHEMIC will provide more or less workers, and the Polymorphic Adaptation feature, will guarantee that the workers, of different hardware types, will best fit to the simulation type and targets in order to find an optimal cost/performance ratio.

For this use-case some basic requirements (Table 3) of MORPHEMIC are very important: specifically, for the simulation tasks, performances are very important, so the support for HPC (MOR-SE.7) and for hardware acceleration

(MOR-SE.8) are requested; moreover, the support for containers (MOR-SA.7), including both Docker and Singularity and serverless application form (MOR-SA.3) is requested as well. Concerning the latter feature, since it includes the explicit request of a technology, Singularity, will be also included among the specific requirements of this use-case. Table 11 contains the requirements provided by CFD use-case, including also the common ones (that can be identified through the ID syntax defined in section 1.3.

Table 11 MORPHEMIC requirements related to CFD use-case

ID	Name	Description	Topic	Priority (MoSCoW)
UC-C-SE.3	Support for GPU	MORPHEMIC could support software or hardware solutions using GPU	Supported Environments	COULD
UC-C-UF.2	Targeted deployment: price	MORPHEMIC should determine and enforce a deployment configuration according to the requirements of price	Parameters for the Utility Function	SHOULD
UC-C-UF.3	Targeted deployment: packaging	MORPHEMIC should determine and enforce a deployment configuration according to the requirements of packaging (i.e., container)	Parameters for the Utility Function	SHOULD
UC-C-UF.5	Targeted deployment: computing power	MORPHEMIC could determine and enforce a deployment configuration according to the requirements of computing power	Parameters for the Utility Function	COULD
UC-3-SE.1	Connection to HPC centres	MORPHEMIC should support connections to already existing HPC centre	Supported Environments	SHOULD
UC-3-SE.2	Management of workers	MORPHEMIC must provide the option to shutdown workers if no longer needed, via metric or direct API call	Supported Environments	MUST
UC-3-SA.1	Targeted deployment: containers	MORPHEMIC must support at least one between Docker and Singularity containers	Supported Application Form	MUST
UC-3-AD.1	Adaptation of the number of workers	MORPHEMIC must dynamically define and use the minimum size of workers needed, even zero workers	Adaptation	MUST
UC-3-UF.1	Targeted deployment: memory	MORPHEMIC must be able to determine and enforce a deployment configuration according to the requirements of memory	Parameters for the Utility Function	MUST
UC-3-UF.2	Targeted deployment: deployment time	MORPHEMIC should be able to determine and enforce a deployment configuration according to the requirements of deployment time	Parameters for the Utility Function	SHOULD
UC-3-SH.1	Track worker velocity	MORPHEMIC must provide a mechanism to record the worker's velocity intended as	Self-Healing	MUST

	ratio between the number of iterations and the elapsed time for a certain worker

3.3.3.1 *Requirements group: deployment environment type*

In order to efficiently run the services of CFD use-case, the possibility to deploy different components on different environments is needed. The requirements UC-C-SE.1, UC-C-UF.3 and UC-3-SE.1 describe the environments that should be supported for this use-case, both in terms of deployment and interaction capabilities. Concerning the application forms, the capabilities to use different types of containers (UC-3-SA.1) or serverless form (MOR-SA.3 - Table 3) are considered very important. This aspect is also part of the basic definition of MORPHEMIC (MOR-SA.2 - Table 3), but in this context, the types of containers to be supported are well specified.

3.3.3.2 *Requirements group: deployment environment's features*

The main features to be taken into account for deploying CFD use-case are price (UC-C-UF.2), memory (UC-3-UF.1) and deployment time (UC-3-UF.2). Deployment constraints such as support for hardware acceleration (MOR-SE.8 - Table 3), GPU (UC-C-SA.3) and computing power (UC-C-UF.5) are not mandatory but useful. Finally, a set of requirements concerns the management of the workers (UC-C-SE.2, UC-C-AD.1 and UC-C-3-SH.1): generally speaking, an infrastructure management platform is always able to manage nodes. However, this aspect is included into the section, since it is considered very important by ICON who expects some benefits from this functionality.

3.3.4 Technological requirements

At time of writing this document some technical aspects are still not consolidated, especially for low-fi components. However, two points have high priority:

- High memory usage (up to 32GB RAM required)
- Multi-node configuration, including data access by the resources.

The deliverable 6.3, due for M12, will include all the details for this use-case (and the others).

4 Fulfilment metrics of the requirements

This chapter provides a recap of the requirements discussed in the previous chapters and proposes a preliminary evaluation method. In particular the evaluation will take into account all the exposed requirements except for the group derived from MELODIC (Table 1 and Table 2), since they are supposed to be already met. The proposed method is focused on the *fit criterion*, which will suggest, for each requirement, a formal *question* to establish unambiguously whether it is fulfilled or not. Finally, section 4.2 will report a comparison between the requirements and the preliminary architecture described in the DoW. This will give a first rough idea about what is already on track, i.e., which functionalities imagined at proposal time are still considered valid after a first evaluation against the use-case and which are the possible improvements that, after this analysis, have been envisioned.

4.1 Requirements and fit criteria

Table 12 contains the requirements obtained by a preliminary analysis of the functionalities of MORPHEMIC (Table 3)
and all the requirements obtained from the use-cases (Table 6, Table 8 and Table 11). The column *Origin* keeps track of all the *originators* and *priority* reports the highest one. The column *Topic* tries to categorize each requirement basing on the main pillars on which MORPHEMIC will be based. Specifically, each requirement represents one of the elements of themes on which MORPHEMIC will be designed. The themes are:

- **Adaptation**, Proactive and Polymorphic
- **Application Modelling**, i.e., to find deployment patterns for a certain application category
- **Audit**
- **Data migration**
- **Optimization**
- **Utility Function** and the parameters to maximize Utility
- **Security**
- **Self-Healing**
- **Supported Application Forms**
- **Supported Environments.**

Most of these themes have been discussed in this deliverable. They are mapped in the Work Packages and in the preliminary architecture. In particular, the mapping between requirements and architecture will be discussed in section 4.2.
Finally, the table includes the *fit criterion*, that will help to define unambiguously each requirement and to evaluate if it will be fulfilled or not. Such criteria will be defined in details in the evaluation strategy (D6.2) and included in the final evaluation (D6.5) to verify if the requirements have been met.

Table 12 Complete list of requirements

ID	Name	Origin	Description	Topic	Prority	Fit Criterion
MOR-SE.1	Polymorphic Environments: Cloud	MORPHEMIC	MORPHEMIC must support Cloud	Supported Environments	MUST	MORPHEMIC shall support the deployment on two different Cloud infrastructures at least
MOR-SE.2	Polymorphic Environments: Hybrid Clouds	MORPHEMIC	MORPHEMIC must support Hybrid Clouds	Supported Environments	MUST	MORPHEMIC shall support the deployment on Hybrid Clouds environment
MOR-SE.3	Polymorphic	MORPHEMIC	MORPHEMIC	Supported	MUST	MORPHEMIC

	Environments: Multi-Cloud	MIC	must support Multi-Cloud environment	Environments		shall support the deployment on Multi-Clouds
MOR-SE.4	Polymorphic Environments: Fog	MORPHE MIC	MORPHEMIC must support Fog	Supported Environments	COULD	MORPHEMIC shall support the deployment on Fog environment
MOR-SE.5	Polymorphic Environments: Edge	MORPHE MIC	MORPHEMIC must support Edge	Supported Environments	MUST	MORPHEMIC shall support the deployment on Edge machines
MOR-SE.6	Polymorphic Environments: bare metal	MORPHE MIC	MORPHEMIC must support on-premises environment	Supported Environments	COULD	MORPHEMIC shall support the deployment on real (non-virtual) machines
MOR-SE.7	Polymorphic Environments: HPC	MORPHE MIC	MORPHEMIC must support HPC	Supported Environments	MUST	MORPHEMIC shall support the deployment on HPC environment
MOR-SE.8	Polymorphic Environments: hardware accelerators	MORPHE MIC	MORPHEMIC must support hardware accelerators	Supported Environments	MUST	MORPHEMIC shall support hardware accelerators
MOR-SE.9	Polymorphic Environments: FPGA	MORPHE MIC	MORPHEMIC must support FPGA	Supported Environments	MUST	MORPHEMIC shall support FPGA
MOR-SE.10	More polymorphic environments	MORPHE MIC	MORPHEMIC could support environments different than the ones listed in MOR-SE.1-MOR-SE.9	Supported Environments	COULD	MORPHEMIC shall support each new environment added
MOR-SA.1	Polymorphic application forms: VM	MORPHE MIC	MORPHEMIC must support Virtual Machines	Supported Application Forms	MUST	MORPHEMIC shall support the deployment onVirtual Machines
MOR-SA.2	Polymorphic application forms: containers	MORPHE MIC	MORPHEMIC must support containers	Supported Application Forms	MUST	MORPHEMIC shall manage containers (i.e., Docker)
MOR-SA.3	Polymorphic application forms: serverless	MORPHE MIC	MORPHEMIC must support serverless application form	Supported Application Forms	MUST	MORPHEMIC shall support serverless application form
MOR-SA.4	More polymorphic application forms	MORPHE MIC	MORPHEMIC could support application form different than the ones listed in MOR-SA.1-	Supported Application Forms	COULD	MORPHEMIC shall support each new application forms added

			MOR-SA.3			
MOR-CON.1	Pre-configure multiple deployment configurations	MORPHEMIC	MORPHEMIC must support multiple deployment configurations in order to plan the deployment for different contexts	Adaptation	MUST	MORPHEMIC shall pre-configure different deployment configuration for the same application
MOR-SH.1	Real time infrastructure performance monitoring	MORPHEMIC	MORPHEMIC must be able to monitor in real time the health status of the infrastructure, in order to use the obtained data for Proactive Adaptation	Self-Healing	MUST	MORPHEMIC shall provide a monitoring method to get health information on the status of the managed infrastructures. Specifically, if the infrastructure is up, down and, at least, critical issues
MOR-SH.2	Real time applications performance monitoring	MORPHEMIC	MORPHEMIC should be able to monitor in real time the health status of the applications, in order to use the obtained data for Proactive Adaptation	Self-Healing	SHOULD	MORPHEMIC shall monitor the status and the performance of the applications
MOR-SH.3	Self-Healing mechanism	MORPHEMIC	MORPHEMIC must support a Self-Healing mechanism able to detect the violation of service level objectives or any other dangerous situations, concerning the deployed application components, in order to enact new adaptations.	Self-Healing	MUST	MORPHEMIC shall provide a monitoring system to detect as soon as possible the violation of service level objectives and an enacting system to rect.
MOR-AD.1	Proactive Adaptation	MORPHEMIC	MORPHEMIC must be able to adapt the deployment configuration (environment and	Adaptation	MUST	MORPHEMIC shall provide and apply alternative solutions concerning the

			application form) according to the predicted needs			deployment configuration if the predicted future context requires it
MOR-AD.2	Prediction capabilities on applications	MORPHE MIC	MORPHEMIC should be able to predict potential problematic trends on applications	Adaptation	SHOULD	MORPHEMIC shall provide a prediction method based on the current status to forecast potential problems
MOR-AD.3	Prediction capabilities on infrastructures	MORPHE MIC	MORPHEMIC should be able to predict potential problematic trends on infrastructures	Adaptation	SHOULD	MORPHEMIC shall provide proactive predictions based on the possible future status of the infrastructures
MOR-MD.1	Application crawling	MORPHE MIC	MORPHEMIC should be able to look for open-source software on a set of popular software repositories (such as GitHub, Apache...)	Application Modelling	SHOULD	MORPHEMIC shall provide crawling functionality for open-source code
MOR-MD.2	Application profiling	MORPHE MIC	MORPHEMIC could be able to define application profiles according to patterns found on open-source software classified by functionalities	Application Modelling	COULD	MORPHEMIC shall provide application profiling functionality
MOR-OP.1	Optimization of Resources	MORPHE MIC	Capability to reuse and optimize the usage of cloud resources as much as possible	Optimization	SHOULD	MORPHEMIC shall be able to share the deployment environment and the resource among different applications (instances)
UC-C-SE.1	Support for non-virtualized components	Use-cases 1/2	MORPHEMIC must support hardware components	Supported Environments	MUST	It shall be possible to configure MORPHEMIC to point to specific hardware

						components and use them for deployment
UC-C-SE.2	Multi-site deployment	Use-cases 1/2	MORPHEMIC must support deployment configurations spanning across multiple geographical locations	Supported Environments	MUST	MORPHEMIC shall be able to deploy different components of a single application on different geographical locations
UC-C-SE.3	Support for GPU	Use-cases 2/3	MORPHEMIC could support software or hardware solutions using GPU	Supported Environments	COULD	MORPHEMIC shall be able to select a deployment environment supporting GPU
UC-C-UF.1	Targeted deployment: network capability	Use-cases 1/2	MORPHEMIC could determine and enforce a deployment configuration according to the requirements of network capability	Parameters for the Utility Function	COULD	MORPHEMIC shall be able to process network capabilities (such as throughput) in Camel model or in the Utility Function
UC-C-UF.2	Targeted deployment: price	Use-cases 1/3	MORPHEMIC should determine and enforce a deployment configuration according to the requirements of price	Parameters for the Utility Function	SHOULD	MORPHEMIC shall be able to exploit cost requirements such that it can deduce and enforce cost-effective application deployments.
UC-C-UF.3	Targeted deployment: packaging	All use-cases	MORPHEMIC should determine and enforce a deployment configuration according to the requirements of packaging (i.e., container)	Parameters for the Utility Function	SHOULD	MORPHEMIC shall be able to process “packaging type” in Camel model or in the Utility Function
UC-C-UF.4	Geographical awareness	Use-cases 1/2	MORPHEMIC must support geographical constraints in the deployment configuration	Parameters for the Utility Function	MUST	MORPHEMIC shall enable the description of geographical location requirements as well as location-based objectives

						which can then be mapped to location-aware application deployments
UC-C-UF.5	Targeted deployment: computing power	use-case 1/3	MORPHEMIC must determine and enforce a deployment configuration according to the requirements of computing power	Parameters for the Utility Function	MUST	MORPHEMIC shall be able to process “minimum computing power” in Camel model or in the Utility Function
UC-C-SEC.1	Support for traffic isolation	Use-cases 1/2	MORPHEMIC could determine and enforce deployment configurations supporting traffic isolation	Security	COULD	MORPHEMIC shall be able to select network services supporting traffic isolation
UC-C-SEC.2	Support for secure communica-tions	Use-cases 1/2	MORPHEMIC could determine and enforce deployment configurations supporting secure communications among deployed components (i.e., private networks)	Security	COULD	MORPHEMIC shall be able to select network components supporting secure communication
UC-C-SEC.3	Support for security related applications	Use-cases 1/2	MORPHEMIC could determine and enforce deployment configurations supporting security elements (such as firewalls, IDS etc)	Security	COULD	MORPHEMIC shall be able to deploy security related network elements (i.e., IDS systems)
UC-1-SE.1	Platform awareness: DPDK	use-case 1	MORPHEMIC should support DPDK solutions	Supported Environments	SHOULD	MORPHEMIC shall be able to deploy DPDK solutions
UC-1-SE.2	Platform awareness: CPU pinning	use-case 1	MORPHEMIC should support CPU pinning solutions	Supported Environments	SHOULD	MORPHEMIC shall be able to deploy CPU pinning solutions
UC-1-SE.3	Support for redundancy	use-case 1	MORPHEMIC could be able to determine and	Supported environments	COULD	MORPHEMIC shall be able to determine and

			enforce a redundant deployment configuration based on the use-case requirements concerning reliability			enforce redundant deployment configurations
UC-1-SE.4	Support for Fault Management	use-case 1	MORPHEMIC should be able to deploy Fault Management tools	Supported environments	SHOULD	MORPHEMIC shall be able to deploy tools providing Fault Management capability
UC-1-SE.5	Support for Configuration Management	use-case 1	MORPHEMIC should be able to deploy Configuration Management tools	Supported environments	SHOULD	MORPHEMIC shall be able to deploy tools providing Configuration Management capability
UC-1-SE.6	Support for Accounting Management	use-case 1	MORPHEMIC should be able to deploy Accounting Management tools	Supported environments	SHOULD	MORPHEMIC shall be able to deploy tools providing Accounting Management capability
UC-1-SE.7	Support for Performance Management	use-case 1	MORPHEMIC should be able to deploy Performance Management tools	Supported environments	SHOULD	MORPHEMIC shall be able to deploy tools providing Performance Management capability
UC-1-SE.8	Support for Security Management	use-case 1	MORPHEMIC could be able to deploy Security Management tools	Supported environments	COULD	MORPHEMIC shall be able to deploy tools providing Security Management capability
UC-1-UF.1	Support for low latency in terms of deployment time	use-case 1	MORPHEMIC could determine and enforce production-ready deployment configurations with minimal deployment latency	Parameters for the Utility Function	COULD	Deployment latency test after having set the requested deployment option
UC-1-UF.2	Targeted deployment: latency	use-case 1	MORPHEMIC could determine and enforce a	Parameters for the Utility Function	COULD	MORPHEMIC shall be able to specify and

	between the deployed components		deployment configuration according to the requirements of the latency in communication between the components			consider latency requirements in the context of application deployments.
UC-1-AD.1	Dynamic deployment configuration	use-case 1	MORPHEMIC must dynamically re-determine the deployment configuration to react to changing situations	Adaptation	MUST	MORPHEMIC shall support application reconfiguration
UC-1-AD.2	Live migration	use-case 1	MORPHEMIC could apply the new determined deployment configuration minimizing the downtime of the service	Adaptation	COULD	MORPHEMIC shall be able to dynamically and transparently modify the deployment configuration
UC-2-SE.1	Support for ProActive	use-case 2	MORPHEMIC must be able to deploy ProActive	Supported environments	MUST	MORPHEMIC shall be able to deploy ProActive
UC-2-SE.2	Support for private execution environment	use-case 2	MORPHEMIC must be able to deploy on private execution environments (including private cloud)	Supported environments	MUST	MORPHEMIC shall support on-premises solutions
UC-2-SE.3	Support for Master orchestrator	use-case 2	MORPHEMIC must be able to deploy master orchestrator components on public execution environment	Supported environments	MUST	MORPHEMIC shall be able to deploy a generic master orchestrator application on at least one of the associated public environments
UC-3-SE.1	Connection to HPC centres	use-case 3	MORPHEMIC should support connections to already existing HPC centre	Supported Environments	SHOULD	The application deployed through MORPHEMIC shall be able to successfully connect to at least one external HPC centre

UC-3-SE.2	Management of workers	use-case 3	MORPHEMIC must provide the option to shutdown workers if no longer needed, via metric or direct API call	Supported Environments	MUST	MORPHEMIC shall be able to successfully manage the workers used in ICON's use-case
UC-3-SA.1	Targeted deployment: containers	use-case 3	MORPHEMIC must support at least one between Docker and Singularity containers	Supported Application Form	MUST	MORPHEMIC shall manage at least one between Docker and Singularity
UC-3-AD.1	Adaptation of the number of workers	use-case 3	MORPHEMIC must dynamically define and use the minimum size of workers needed, even zero workers	Adaptation	MUST	MORPHEMIC shall dynamically increase or decrease the number of workers for the simulations of ICON's use case according to the needs
UC-3-UF.1	Targeted deployment: memory	use-case 3	MORPHEMIC must be able to determine and enforce a deployment configuration according to the requirements of memory	Parameters for the Utility Function	MUST	Verification of the possibility to include memory as constraint in Camel model or as input in the Utility Function
UC-3-UF.2	Targeted deployment: deployment time	use-case 3	MORPHEMIC should be able to determine and enforce a deployment configuration according to the requirements of deployment time	Parameters for the Utility Function	SHOULD	MORPHEMIC shall be able to process "deployment time" in Camel model or in the Utility Function
UC-3-SH.1	Track worker velocity	use-case 3	MORPHEMIC must provide a mechanism to record the worker's velocity intended as ratio between the number of iterations and the	Self-Healing	MUST	MORPHEMIC shall record the workers velocity and the results should be accessible

elapsed time for a certain worker

The requirements in Table 12 cover a wide range of different functionalities: the use-cases in this sense have been very useful to clarify some aspects and extend some preliminary concepts.

The deliverable 6.2 will define the *Validation Framework* to evaluate how the outcome of the project is suitable to the needs of the use-cases. That evaluation in this sense will be focused on KPIs defined on the DoW and by the detailed definition of the use-cases. On the other hand, most of the aforementioned requirements define specific functionalities and their evaluation is the response to the *fit criteria*: YES or NO.
These requirements and their fit criteria will be taken into account for each component of MORPHEMIC. The final evaluation of the outcome will strongly depend on the number of YES obtained.

4.2 Preliminary evaluation against the architecture

Having a complete list of requirements (Table 12) it is possible to compare the preliminary architecture, shown in Figure 3 MORPHEMIC architecture tentative,with them. The better understanding of the use-cases and the deeper requirement analysis performed during the first months of the project offer the possibility to review the architectural schema, to understand if the direction is right, should be integrated or is wrong at all. This will be a preliminary evaluation of the performed work and a deeper clarification of the needed functionalities. Furthermore, the requirements will help to better define the details of the components, and will not end with this document. The process of refining will proceed in parallel for use-cases, architecture and requirements in the next months: it could also be possible that some requirements define new functionalities to be mapped in new components. This section reports this analysis.
Figure 20 shows the preliminary architecture already reported in Figure 3 MORPHEMIC architecture tentative. Core of the architecture is MELODIC: its requirements (Table 1 and Table 2) will be extended to support *Polymorphic Adaptation* in terms of *deployment environment* and *application form*.

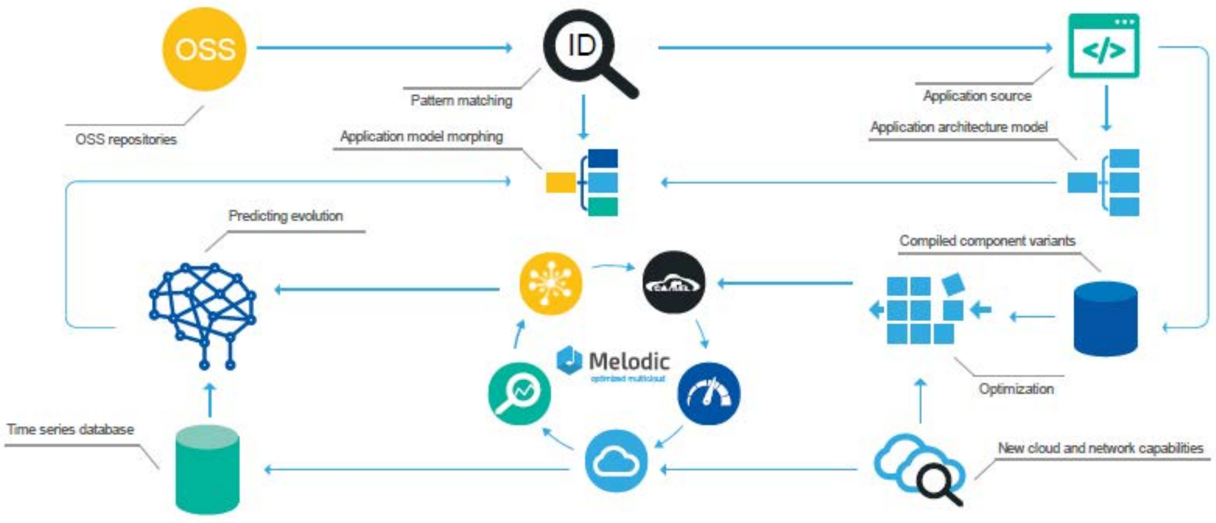


Figure 20 Preliminary architecture of MORPHEMIC

Table 13 shows the subset of requirements the features added to ones provided by MELODIC. This set integrates the ones on Table 1 and Table 2: it mainly includes Polymorphic Adaptation along with new environments and application forms to be supported.

Table 13 Requirements and components related to the extension of MELODIC

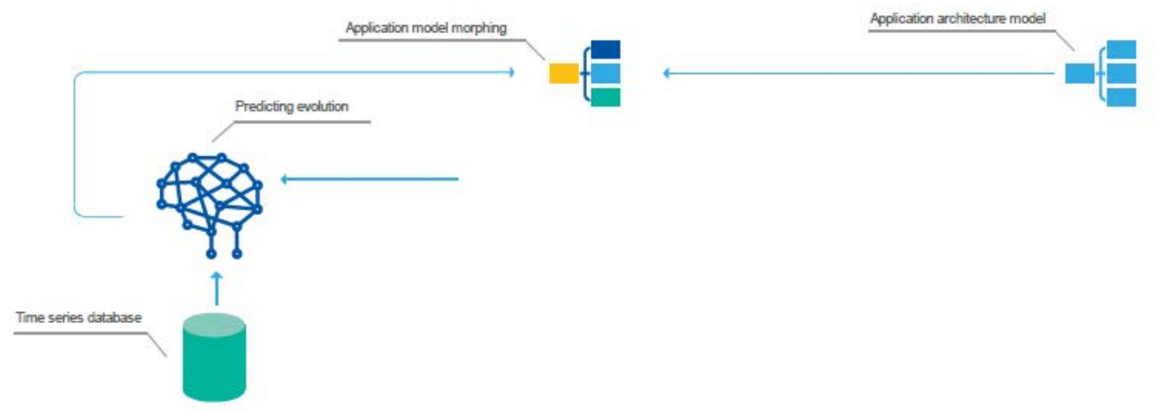
	ID	Name
	MOR-SE.1	Polymorphic Environments: Cloud
	MOR-SE.2	Polymorphic Environments: Hybrid Cloud
	MOR-SE.3	Polymorphic Environments: Multi-Cloud
	MOR-SE.4	Polymorphic Environments: Fog
	MOR-SE.5	Polymorphic Environments: Edge
	MOR-SE.6	Polymorphic Environments: bare metal
	MOR-SE.7	Polymorphic Environments: HPC
	MOR-SE.8	Polymorphic Environments: hardware accelerators
	MOR-SE.9	Polymorphic Environments: FPGA
	MOR-SE.10	More polymorphic environments
	MOR-SA.1	Polymorphic application forms: VM
	MOR-SA.2	Polymorphic application forms: containers
	MOR-SA.3	Polymorphic application forms: serverless
	MOR-SA.4	More polymorphic application forms
	UC-C-SE.1	Support for non-virtualized components
	UC-C-SE.2	Multi-site deployment
	UC-C-SE.3	Support for GPU
	UC-C-UF.1	Targeted deployment: network capability
	UC-C-UF.2	Targeted deployment: price
	UC-C-UF.3	Targeted deployment: packaging
	UC-C-UF.4	Geographical awareness
	UC-C-UF.5	Targeted deployment: computing power
	UC-C-SEC.1	Support for as traffic isolation
	UC-C-SEC.2	Support for secure communications
	UC-C-SEC.3	Support for security related applications
	UC-1-SE.1	Platform awareness: DPDK
	UC-1-SE.2	Platform awareness: CPU pinning
	UC-1-SE.3	Support for redundancy
	UC-1-SE.4	Support for FCAPS
	UC-1-SE.5	Support for Fault Management
	UC-1-SE.6	Support for Configuration Management
	UC-1-SE.7	Support for Accounting Management
	UC-1-SE.8	Support for Performance Management
	UC-1-UF.1	Support for low latency in terms of deployment time
	UC-1-UF.2	Targeted deployment: latency between the deployed components
	UC-1-AD.1	Dynamic deployment configuration
	UC-1-AD.2	Live migration
	UC-2-SE.1	Support for ProActive
	UC-2-SE.2	Support for private execution environment
	UC-2-SE.3	Support for Master orchestrator components on public environment
	UC-3-SE.1	Connection to HPC centres
	UC-3-SE.2	Management of workers
	UC-3-SA.1	Targeted deployment: containers

	UC-3-UF.1	Targeted deployment: memory
	UC-3-UF.2	Targeted deployment: deployment time

Most of the functionalities concerning *Polymorphic Adaptation* is implemented by meeting the requirements of this group. Specifically, a new version of MELODIC will perform similar operations for a wider set of use-cases: this new version will represent the core of a platform that will support Fog, Edge and real machines besides the Cloud. The environments and application forms will expose more options to support very specific functionalities, related to networking (5G Cloud-RAN use-case), HPC or strict privacy.

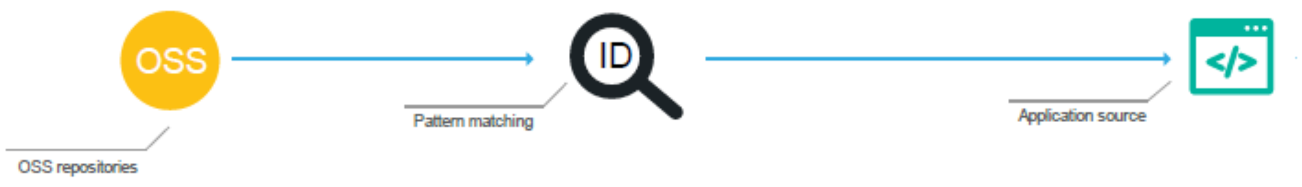
Table 14 concerns the requirements related to Proactive Adaptation. Apparently, this set is not requested by any use-case. However, looking at the benefit tables (Table 5 and Table 7), it is clear that at least 5G Cloud RAN and E-Brain Science use-cases consider it as a benefit. The objective of Proactive Adaptation is to reduce the overhead of re-configuration by planning and executing it in advance with respect to the moment when the need is verified. This will result in a improvement in terms of performance of the overall application. It is possible that also CFD use-case, where performances are critical, will benefit from this aspect.

Table 14 Requirements and components for Proactive Adaptation



ID	Name
MOR-CON.1	Pre-configure multiple deployment configurations
MOR-AD.1	Proactive Adaptation
MOR-AD.2	Prediction capabilities on applications
MOR-AD.3	Prediction capabilities on infrastructures
MOR-SH.1	Real time infrastructure performance monitoring
MOR-SH.2	Real time applications performance monitoring
MOR-SH.3	Self-Healing mechanism
UC-3-AD.1	Adaptation of the number of workers
UC-3-SH.1	Track worker velocity

Table 15 Requirements and components for application profiling



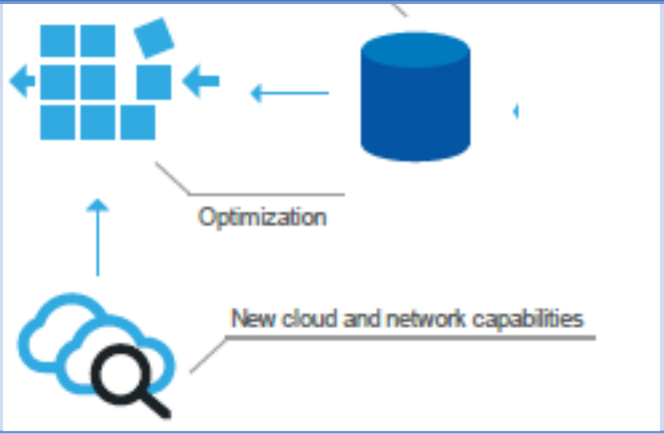
ID	Name
MOR-MD.1	Application crawling

MOR-MD.2	Application profiling

Application profiling is an important aspect of Polymorphic Adaptation, Table 15 shows the requirements strictly related to this functionality. The picture shows the flow of this feature: a set of repositories are crawled by a specific component that gets the data of projects stored on them. These projects are categorized according to their functionalities and, for each resulting category, specific *patterns* are retrieved in order to define *application profiles*. Objective of this functionality is to retrieve a set of deployment models from the application profiles in order to define deployment patterns for specific application categories. This will make easier polymorphic and Proactive Adaptation by suggesting deployment models suitable for the application. For the final user this will have two main advantages:

- propose deployment configuration with high possibility to be optimal for the requested usage
- prepare deployment configuration optimal for future contexts (Proactive Adaptation.

Table 16 Requirements and components for optimization

	ID	Name
	**MOR-OP.1**	Optimization of Resources

The last identified requirement concerns the optimal usage of resources. Table 16 shows the components that will provide this functionality, focused on Cloud but potentially extensible in other contexts.

At time of writing this document the use-cases and some aspects of the MORPHEMIC's functionalities are still not fully consolidated. For example, some use-case partners would like to use MORPHEMIC to extend their applications, better exploiting functionalities that can be provided to the customers. Detailed technological aspects are actively under discussion from several points of view: often technological, but sometimes also functional. This means that the presented requirements should be considered as a very consistent and meaningful starting point. Some of them will be refined and detailed: others will be fully adopted by the use-cases that will become more and more aware of their *implicit* requirements. Maybe some other requirements will be added to include functionalities still not identified or to detail features still not clear.
In any case this deliverable will be used as starting point for the definition of the architecture and for the evaluation of the final result: the *fit criteria* of the requirement will be the first validation method proposed for MORPHEMIC. Deliverable 6.2 will provide the others.

5 Conclusions

This deliverable is the outcome of the work on two aspects:

- the detailed analysis of the concept, objectives and functionalities to be provided by MORPHEMIC
- the analysis of the use-cases which provided clear ideas of the needs of different application domains and different markets.

Each aspect defined a set of requirements. The two sets are consistent and integrate each other, well defining the functionalities that MORPHEMIC will provide.
MELODIC, the consolidated starting point, represented a big help in the analysis, providing a group of preliminary requirements that completely defined all the aspects related to Cloud infrastructure management. Furthermore, its potential improvements, on which the MORPHEMIC proposal is based, helped to define and analyse the requirements related to the other deployment environments (For, Edge, real machines...) and application forms (containers, serverless...).
In turn, each use-case provided more information related to a specific domain. Software-defined networks, related to 5G Cloud-RAN, need deployment configurations with special care on latency. Security is the most critical factor for healthcare information managed by E-Brain science use-case. Memory, computing power and deployment time, that may be reduced through Proactive Adaptation, are the features more requested by Computational Fluid Dynamics.
All the use-cases will benefit from polymorphic and proactive aspects: the benefits will affect different tasks in different ways but will be an important added value for the business of each use-case partner.
This added value will be expressed by the *Utility* which, in turn, will be used to maximize the benefits of the chosen deployment configuration for each use-case at any moment. The dynamic modification of the deployment will be evaluated by making a trade-off between the costs, in terms of unavailability or partial availability, and benefits and, if the overall Utility increases, the deployment configuration will be immediately modified.
This use-cases also contributed to ensuring that the workplan imagined at proposal time is correct. Specifically, the starting point, MELODIC, is useful and provides several functionalities needed by the considered application domains. The planned extensions, in the direction of Polymorphic and Proactive Adaptation, are consistent with the needs of the markets that the use-cases represent. Definitely, the preliminary evaluation of the concept of MORPHEMIC demonstrates that the idea is well conceived and suitable for the real-life applications proposed by the use-cases partners.
Finally, the use-cases will obtain as more benefits as more requirements will be fulfilled by the complete the MORPHEMIC platform. The fit criteria associated with the requirements give a quick and unambiguous idea about the fulfilment of each of them. This simple evaluation method will be used for the whole project.
The deliverable 6.1 is part of the work that the Consortium intends to do for ensuring strong cooperation between the use-cases partners and the technological partners. This work will include, along with the industrial requirements, also the support on the use –cases deployment and the evaluation. Since the domains of the use-cases cover a wide range of applications, this cooperation will represent an important added value in terms of exploitation. The next steps of the Work Package 6 will be:

- detailed definition of the use-cases (Deliverable 6.3 - *Use Case definition and preparation* - M12), which may raise new requirements
- implementation of the use-cases (Deliverable 6.4 - *Use-cases prototypes* – M32)
- definition of the validation framework (Deliverable 6.2 – *Validation Framework Design* – M34), which will include more details on the evaluation: that evaluation will also concern the KPIs associated with the SLA needed by the use-cases
- the result of the whole process (Deliverable 6.5 - *Validation Results* – M36).

In general, D6.1 will be a reference for the whole project and will be integrated by the contributions that will be produced in the next months, when the analysis will clarify more aspect, define new functionalities and consolidate the concept of the platform.

6 References

[1] A. P. Achilleos *et al.*, ‘The cloud application modelling and execution language’, *J. Cloud Comput.*, vol. 8, no. 1, p. 20, Dec. 2019, doi: 10.1186/s13677-019-0138-7.

[2] Yiannis Verginadis, Geir Horn, Kyriakos Kritikos, Feroz Zahid, Daniel Baur, Paweł Skrzypek, Daniel Seybold, Marcin Prusiński, Somnath Mazumdar, 'Melodic Deliverable 2.2 Architecture and initial feature definitions'. [Online]. Available: https://melodic.cloud/wp-content/uploads/2019/01/D2.2-Architecture-and-Initial-Feature-Definition.pdf.

[3] Yiannis Verginadis, Wiktor Zolnieroiwcz, Paweł Skrzypek, Daniel Seybold, Kyriakos Kritikos, Somnath Mazumdar, Antonia Schwichtenberg, Feroz Zahid, Jörg Domaschka, Geir Horn, Ernst Gunnar Gran, Daniel Baur, Hynek Masata, Paweł Gora, 'Melodic, Deliverable 2.1 System specification document'. [Online]. Available: https://melodic.cloud/wp-content/uploads/2018/11/D2.1-System-Specification.pdf.