Melodic
Big data cloud

Multi-cloud Execution-ware for Large-scale Optimized Data-Intensive Computing

H2020-ICT-2016-2017
Leadership in Enabling and Industrial Technologies; Information and Communication Technologies
Grant Agreement No.:
731664

Duration:
1 December 2016
30 November 2019
www.melodic.cloud

Deliverable reference:
6.1

Date:
28 February 2018

Responsible partner:
CAS
Editor(s):
Sébastien Kicin

Author(s)
Sébastien Kicin, Sebastian Schork, Antonia Schwichtenberg, Geir Horn, Paweł Gora, Tomasz Przeździęk, Michal Semczuk

Approved by:
Jörg Domaschka

ISBN number:
N/A

Document URL:
http://www.melodic.cloud/deliverables/D6.1 Evaluation Framework and Use Case Planning.pdf

Title:

Evaluation Framework and Use Case Planning

Abstract/Executive summary

The evaluation framework is used to guide the evaluation of MELODIC, and mechanisms integrated in the different use cases have been defined. In the scope of each use case, specific evaluation scenarios for validating and demonstrating the MELODIC platform are documented.

The evaluation framework is based on four phases inspired by the Goal/Question/Metric (GQM) method: the planning phase, the definition phase, the data collection phase, and the interpretation phase. It will lead to an integration of evaluation criteria of qualitative (e.g. feedback from business managers) and quantitative natures (e.g. deployment performance measurements).

The test case scenarios developed in the scope of WP5 have been used as a baseline while building the specification of each evaluation scenario. A utility features method with concrete examples has also been defined.

The framework identifies appropriate evaluation methods for the qualitative and quantitative data collection and analysis and creates as a whole an 'evaluation tool box' for the execution of the use case specific scenarios. Partners collaboratively defined a set of evaluation scenario activities and analysed what need to be implemented. This resulted in in-detail organization and planning of the use case demonstrations.

The targeted milestone M4 will mark the application of the initial MELODIC platform in the use cases, and first round of feedback on its use.

This deliverable should be read by the following persons:

1. Evaluation team – to verify the validity of the described evaluation framework and plan,
2. Development teams - to confirm the overall feasibility of the use cases and of the related evaluation based on the MELODIC platform release plan.

The deliverable reflects the work done in the scope of WP6 (T6.1) until February 2018.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Document	
Period Covered	M7-15
Deliverable No.	D6.1
Deliverable Title	Evaluation Framework and Use Case Planning
Editor(s)	Sébastien Kicin
Author(s)	Sébastien Kicin, Sebastian Schork, Antonia Schwichtenberg, Geir Horn, Paweł Gora, Tomasz Przeździęk, Michal Semczuk
Reviewer(s)	Daniel Seybold, Amir Taherkordi
Work Package No.	6
Work Package Title	Market driven use cases and validation
Lead Beneficiary	CAS Software AG
Distribution	PU
Version	1.0
Draft/Final	Final
Total No. of Pages	112

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Table of Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Index of Figures

Index of Tables

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

1 Introduction

The deliverable reflects the work done in the scope of WP6 (T6.1) until February 2018. Melodic needed an efficient evaluation methodology specifying how to obtain feedback from Melodic use case partners.
Melodic is not the first project raising performance assessment questions of a software platform. In the past, each ICT project developed under European EC support has been confronted with the same question: how should the added value, performance and reliability of IT-based prototypes be assessed? The challenge of this assessment has been actually derived from the fact that IT systems are usually developed as interfaces between different evaluation aspects. Four aspects have been identified in relation to the demonstration of the Melodic platform (see figure 1).

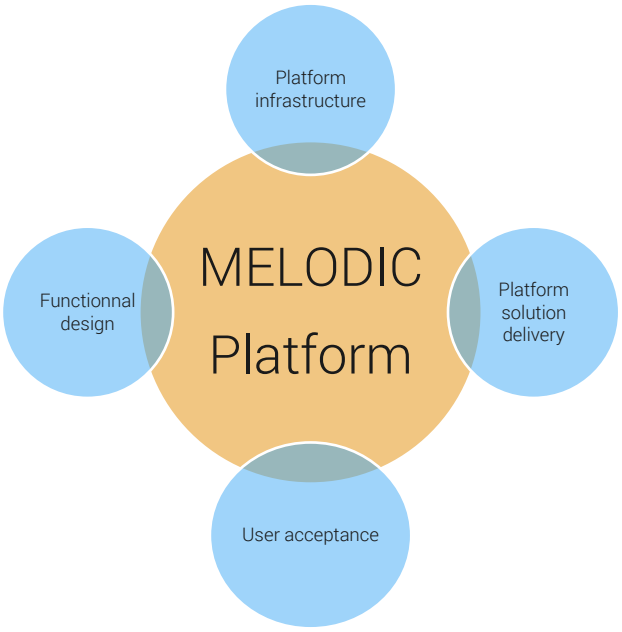


Figure 1 Melodic evaluation aspects

User acceptance is the core aspect of Melodic evaluation at this stage of the project in order to be able to prepare the initial business deployment of the Melodic platform beyond the end of the project. The functional design needs to be evaluated in detail in order to ensure the practical relevance of the features developed so far and check their conformity with functional and non-functional requirements of the use cases and with the specified requirements of the Melodic platform (see project deliverable D2.1).

The platform solution delivery as such (documentation, support, etc.) is not currently considered as a high evaluation priority as we are dealing first with a system prototype. The infrastructure includes both the Melodic components and the actual resources where Melodic is running (e.g. OpenStack UULM, OpenStack UiO, and ArubaCloud). This infrastructure is clearly scalable and

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

will constitute a much more relevant issue while starting the business deployment phase of the Melodic solution, even if basic feedback could be collected at this stage.

1.1 Scope of the Document

This document presents the Melodic evaluation framework and overall evaluation approach that will be implemented in WP6 according to the different Melodic use cases. Based on the Goal/Question/Metric method (GQM), this document is paving the way to the use case implementations towards final evaluation execution by the end of the project. The "test scenarios" already used in WP5 are adapted and consolidated as "evaluation scenarios" in this document. Key technical quality focuses and business priorities are identified in order to prepare the deployment of detailed appropriated evaluation questions and metrics.
The reasoning challenge is also addressed by presenting the way the approach of "utility function" can be used to solve software deployment optimisation.
A preliminary description of the architecture of each use case was provided in detail in the Deliverable D2.1. This document is preparing the related WP6 implementation steps for each use case by specifying major issues like the targeted business impacts, use case roles and evaluation participants.

1.2 Overall Evaluation Objectives

Based on the Melodic use cases, the evaluation should demonstrate the main benefits of the Melodic approach to the European (big data) software industry, including:

- Ability to grow and scale up the applications managed by Melodic through multi-cloud deployments;
- Possible lower cloud costs due to countering vendor lock-in;
- Optimised efficiency of computation on big data sets dispersed over multiple physical locations including data location management;
- ability of automatic reconfiguration which enables an application to be more robust and achieve the delivery of stable service level;
- cross-cloud ability allowing configuration of geographical location;
- less management effort and cost and a higher automation level for adaptive application provisioning.

Applicability to different scenarios and wider class of problems is the key to receive traction and acceptance from software professionals and businesses, and attract them.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Each of the industrial consortium partners provides at least one use case scenario and will execute the related demonstrator to present how Melodic can be used by a variety of organisations in different contexts. A preliminary description of the architecture of each use case was provided in detail in Deliverable D2.1. The most typical scenarios explaining the exploitation model of Melodic will include:

a) Data-intensive application owners seeking to scale-up through the app market distribution model (use case 1 by CAS);
b) Data-intensive application providers trying to enhance/upgrade their offering (use case 2 by CET);
c) Software providers looking for ways to multiply and integrate the services offered to their customers (use-cases 3 and 4 by 7bulls).

The use case implementation will demonstrate:

- how Melodic can make highly innovative smart city services, relying on sensitive big-data of people and vehicle mobility, economically and technically feasible. In particular, the deployment application will support the intelligent traffic control management system in managing crises and abnormal situations (CET use case)
- how Melodic will allow big genome data be processed for medical purposes in an affordable and highly secure way (7bulls first use case)
- how Melodic can ease, speed up and reduce cost of the deployment of cloud enabled applications distributed by an app store operator (CAS use case)
- how Melodic can help a cloud provider to realise a forward-looking competitive strategy based on offering a value added multi-cloud service (7bulls second use case).

In all of the above use cases, which implementation method can be replicated and scaled up without limits, in Europe and beyond, Melodic will improve application performance and cost effectiveness over a geo-distributed infrastructure.

1.3 The Phases of the Melodic Evaluation

Although there are a number of comprehensive evaluation and validation methodologies in industry and academia, they often lack the goal-driven nature of businesses, thus not able to provide valuable conclusions about the real viability and sustainability of the Melodic platform. From the business perspective, Melodic only provides the means to achieve other business goals, such as deployment process improvement, reduced error rates, and decreased application deployment delays.

The Goal/Question/Metric method (GQM) method [1] supports such a business driven quality improvement and validation approach quite well and has inspired the Evaluation Framework employed for validating the Melodic platform.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

GQM represents a systematic approach for tailoring and integrating goals with models of the software processes, products and quality perspectives of interest, based upon the specific needs of the project. The result of the application of the GQM method is the specification of a strategy targeting a particular set of issues and a set of rules for the interpretation of the measurement data. The principle behind the GQM method is that evaluation, validation and subsequent measurement should be goal-oriented.

Along the GQM method, a certain goal is defined which is refined into questions, and metrics that provide the information to answer these questions. By answering the questions, the measured data can be analysed to identify whether the goals have been attained. Thus, GQM defines metrics from a top-down perspective and analyses and interprets the measurement data bottom-up, as shown in Figure 2.

Using the GQM method, the objects of study are to be clearly identified and then validated according to a number of goals that enable focus on certain aspects of assessment. Each goal will be broken down into one or more questions that act as a vehicle for the assessment of the goal. Finally, in order to analyse and interpret the questions' results, specific metrics will be defined.

The measurement data is interpreted bottom-up. As the metrics are defined with an explicit goal in mind, the information provided by the metrics is interpreted and analysed with respect to this goal, to conclude whether or not it is attained. GQM trees of goals, questions and metrics are usually built based on the knowledge of experts.

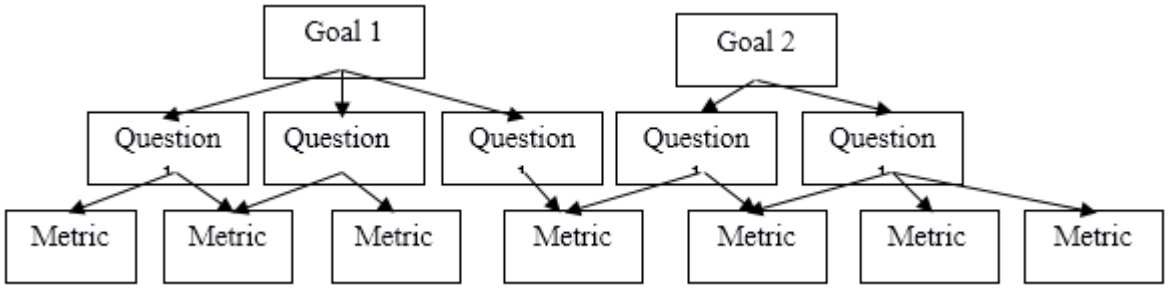


Figure 2 The GQM Method

The Melodic Evaluation Framework contains four phases inspired by the GQM method:

- The *Planning* phase, during which the overall approach is defined and planned, resulting in a use case *evaluation plan*. The evaluation objects are defined (components, processes or resources under observation) as well as the evaluation groups (people who will participate in the evaluation process). This phase is performed to fulfil all basic requirements for conducting the validation successfully, including the definitions of actors, who will be involved, and the creation of a high-level evaluation plan.
- The *Definition* phase, during which the measurement scheme is defined (goal, questions and metrics are defined) and documented.
- The *Data Collection* phase, during which the actual data collection takes place, resulting

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

in collected measurement and data. The data collection forms are defined, filled-in and stored.

- The *Interpretation* phase, during which collected data is processed with respect to the defined metrics into measurement results that provide answers to the defined questions, after which goal attainment can be evaluated.

At this stage of the project, the planning and the definition phase have been achieved. Figure 3 depicts the different phases and their connection.

Figure 3 The overall Melodic evaluation phase

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

2 The Planning Phase

The primary objectives of this phase were to collect all required information for a successful Melodic evaluation, to define the actors involved in the procedure and to prepare a high-level validation framework. This framework is supposed to act as guideline for all subsequent phases and all stakeholders involved.

Three identification steps are involved in the planning phase of the Melodic evaluation:

1. The groups of people who will participate in the evaluation process.
2. The validation perspectives.
3. The objects to be validated.

The following sections describe the preliminary outcome of the planning phase. However, taking into account possible updates in the prototype development (WP5), these results are indicative and subject to further revisions until the actual validation phase performed by the end of the project.

2.1 The Melodic Evaluation Groups

Since the Melodic project is focusing on adaptive provisioning, the evaluation should not only involve the developers in WP3, WP4 and WP5, but also deployment actors and people who will be the actual users of the Melodic platform. Therefore, the Evaluation Framework defines three separate validation roles from the use case partners or from the developers of the Melodic framework:

- The Melodic developer group: they will form the core validation group with thorough and in-depth knowledge about software quality.
- The Melodic business manager group: they will answer questions with a special focus on measurable business impact on adaptive provisioning. This group based on participants from the use-case partners will be involved through a survey on the business impact of the Melodic solution.
- The Melodic administrator group: This group will particularly take care of the technical performance, liability, maintainability, and usability of the platform.

Preliminary lists of participants of evaluation groups for each use case have been identified and are presented in Chapter 4.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

2.2 The Melodic Evaluation Perspectives

Each evaluation object defined in the next section will be validated according to two different perspectives that are defined below:

1. The *Technical Perspective*, in which the quality, liability and technical performance of the Melodic platform will be validated.
2. The *Business Perspective*, in which the response to the user needs and the business impact will be examined. The instrumentation used in this perspective is drawn upon usefulness and feasibility of the proposed deployment operation support.

Administrators and developers will interact directly with the system. Consequently, the procedure used to evaluate the Melodic platform from the technical perspective should only target these two groups.

Table 1 provides an overview of the importance of the perspective according to the evaluation groups.

Table 1 Perspectives according to evaluation groups

Evaluation groups	**Technical perspective**	**Business perspective**
Business Managers		+++
Administrators	++	+
Developers	+++	

In an early phase of the project and in the first iteration of the use case implementation, the technical perspective is given priority whereas in a later phase of the project and during the second iteration of the implementation, the business perspective will become more important.

2.3 The Evaluation Objects

After the definition of the Melodic evaluation groups and evaluation perspective, the next step is the identification and selection of appropriate evaluation objects.

The evaluation of the Melodic platform will cover all the Melodic components:

- CP Generator
- DLMS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

- Esper/Monitoring
- Utility Generator
- Meta Solver
- CP Solver
- LA Solver
- Solver to deployment
- Adapter/Plan Generator
- ESB
- Cloudiator
- REST CLIENT
- CDO Server

Nevertheless, even if all these Melodic platform components are involved in the evaluation process, they cannot be considered as evaluation objects themselves for use case partners, neither on business nor on technical perspectives.
The evaluation objects should mainly draw upon the evaluation groups' needs. The latter are not directly related to the Melodic component architecture.
The partners agreed to consider "use case evaluation scenarios" as the objects of study. In particular, each use case should be based on one or more "evaluation scenario".

An evaluation scenario is based on a "*composition of evaluation components*". Evaluation components are directly related to the test scenarios described in WP5 (see Deliverable D5.04 [2]) and are very similar to them (see evaluation component list in Annex 1). An evaluation scenario is therefore a selection of evaluation components put in a concrete order of execution. Further details (like e.g. the number of components included in the application to be deployed and the number of cloud providers) will be provided by the use case partners in D6.2 in order to get a full description of the use case.
The following abstract and common evaluation scenarios were identified by the Melodic use case partners.

Table 2 Evaluation scenario overview

Id.	Name	Short description
1	Setup Melodic	Initial setup and configuration on a blank VM.
2	Add Application	Includes configuration of CAMEL model, constraints etc.
3	Deploy Application	By using the user interface or the API.
4	Optimisation	Under use of the utility function and the scaling mechanisms of Melodic.
5	Local reconfiguration	Define local reconfiguration rules through the use of defined

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

		scalability rules in the CAMEL model of the application.
6	Undeploy Application	By using the user interface or the API.
7	Template-based Utility Function creation	Evaluation of the editors.
10	Extended Stress Test	Evaluation of the capability to scale.
11	Backup and Recovery of Melodic	e.g., moving Melodic to another VM.
12	Monitor system status	Monitoring of the deployment as it is usually done in server/cloud environments by both developers and IT administrators (deployment status, application status, platform status, etc.).
13	Platform Security	Validates various security aspects of the Melodic platform.

These scenarios are detailed in the following chapters including a description of the process and the related selected evaluation components for each of them. Not selected evaluation components are also clearly identified. Some of the scenarios need to be evaluated sequentially as they have dependencies. Detailed references to "test scenarios" presented in D5.04 are provided as far as available.

2.3.1 Evaluation Scenario 1: Setup Melodic

Melodic shall be installed on a blank machine and afterwards configured according to the partner's individual needs. The administrator will perform the installation according to the manuals and documentation provided by the technical partners. Melodic shall be considered "setup" as soon as installation test scripts confirm that the correct setup and initial configuration have been accomplished (e.g., adding users, IPs, etc.).

Table 3 Evaluation components of scenario 1

Topic/Evaluation component	**Selected component (X)**	**ID (according to D5.4)**
User management		
Adding user	X	8.1
Removing user		8.2
Updating user password	X	8.3
Updating user profile	X	8.4
Unified administration procedure		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Unified starting, stopping and restarting of Melodic platform	X	8.5
Configuring backup		8.6
Executing backup		8.7
Recover Melodic platform		8.8
Monitor Melodic platform		8.9

2.3.2 Evaluation Scenario 2: Add Application

An application shall be added to a Melodic Installation. The (authenticated) evaluating person uses the UI provided by Melodic to perform all necessary steps:

- definition of CAMEL model for the application
- making binaries of the application available
- storing the CAMEL model in the platform.

This scenario does not include the deployment of the application itself. The scenario can be considered 'executed', when the application is 'setup' and deployable.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Table 4 Evaluation components of scenario 2

Topic/Evaluation component	Selected component (X)	Test scenario ID (see D5.04)
API		
Camel model upload	X	N/A
Initiate deployment process		N/A
Get application status		N/A
UI		
Web based UI for application view: Application view	X	N/A
Web based UI for application view: Deployment view		N/A
Eclipse based editor for the CAMEL model: CAMEL Model validation	X	N/A
Eclipse based editor for the CAMEL model: Syntax completion	X	N/A

2.3.3 Evaluation Scenario 3: Deploy Application

An application shall be deployed by a Melodic installation. The (authenticated) evaluating person (or entity) uses the UI (or API) provided by Melodic to perform all necessary steps in order to get an already existing application deployed:

The application is considered deployed when Melodic confirms the deployment (i.e. by checking the UI and Monitoring) and the application (and components) is (are) up and accessible (application specific verification).

Table 5 Evaluation components of scenario 3

Topic/Evaluation component	Selected component (X)	Test scenario ID (see D5.04)
Initial deployment		
Installation and deployment of a N-component application on M different Cloud Providers	X	1.3
Installation and deployment of a N-component application in Docker containers on M different Cloud Providers	X	1.6
Installation and deployment of a N-component application, where X components are installed in a Docker container and Y on a normal VM on M different Cloud Providers	X	1.7
Deployment requirements enforcement.	X	1.8

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Installation and deployment of a N-component application on M different Cloud Providers with more advanced set of requirements, like non-functional ones.	X	1.9
Reasoning		
Linear constraints and optimization solving - CP Solver	X	5.1
Linear constraints and optimization solving - MILP Solver	X	5.2
Linear constraints and optimization solving - LA Solver	X	5.3
Non-linear constraints and optimization solving - CP Solver	X	5.4
Non-linear constraints and optimization solving - LA Solver	X	5.5
API		
Camel model upload		
Initiate deployment process	X	N/A
Get application status	X	N/A
UI		
Web based UI for application view: Application view	X	N/A
Web based UI for application view: Deployment view	X	N/A
Eclipse based editor for the CAMEL model: CAMEL Model validation		
Eclipse based editor for the CAMEL model: Syntax completion		
BigData management		
Big data application deployment optimization	X	N/A
Big data application deployment execution	X	N/A
Big data application monitoring and reconfiguration		
Data locality awareness - features related to data locality and data movement.		
Performance		
Response time while solving complex allocation problems	X	6.5
Dynamic scalability within one Cloud - verification of the execution time		6.6
Dynamic scalability testing for multi-Cloud feature (using two different locations)		6.7
Counting Compute Resource Overhead of Melodic introduced over its host machine		6.8

2.3.4 Evaluation Scenario 4: Optimization

Melodic shall handle the reconfiguration of an already deployed and running application according to the non-functional constraints posed as well as the respective optimisation objectives specified in the form of optimising a certain utility function. Since this mechanism is supposed to work (almost) automatically, no actions on the Melodic platform from the evaluating person are needed. Instead, the scenario is triggered by either stressing the deployed application

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

(e.g., in the case of CAS Software AG by using 70-80% of RAM or taking into account more complex rules) or by simulating such circumstances, e.g., by using the API or the message bus. The scenario also contains downscaling in case of an overprovisioned environment for the application.

Table 6 Evaluation components of scenario 4

Topic/Evaluation component	**Selected component (X)**	**Test scenario ID (see D5.04)**
Metric management		
Built-in raw metrics collection	X	2.1
Custom raw metrics collection	X	2.2
Composite metric collection	X	2.3
Event generation	X	2.4
Global reconfiguration		
Attributes of used VM offerings changed	X	4.1
Global reconfiguration	X	4.2
Reasoning		
Linear constraints and optimization solving - CP Solver	X	5.1
Linear constraints and optimization solving - MILP Solver	X	5.2
Linear constraints and optimization solving - LA Solver	X	5.3
Non-linear constraints and optimization solving - CP Solver	X	5.4
Non-linear constraints and optimization solving - LA Solver	X	5.5
API		
Camel model upload		
Initiate deployment process		
Get application status	X	
UI		
Web based UI for application view: Application view		
Web based UI for application view: Deployment view	X	
Eclipse based editor of the CAMEL: CAMEL Model validation		
Eclipse based editor of the CAMEL: Syntax completion		
BigData management		
Big data application deployment optimization	X	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Big data application deployment execution		
Big data application monitoring and reconfiguration	X	
Data locality awareness - features related to data locality and data movement.		

2.3.5 Evaluation Scenario 5: Local reconfiguration

Table 7 Evaluation components of scenario 5

Topic/Evaluation component	**Selected component (X)**	**Test scenario ID (see D5.04)**
Local reconfiguration		
Scale out application	X	3.1
Scale in application	X	3.2

2.3.6 Evaluation Scenario 6: Undeploy Application

An already deployed and running Application shall be undeployed. The (authenticated) evaluating person (or Entity) uses the UI (or API) provided by Melodic to perform all necessary steps in order to undeploy an existing deployed application (manual undeployment triggering on the deployed application)

The application can be considered undeployed when Melodic indicates the application as undeployed and the application is not up and accessible any longer.

Table 8 Evaluation components of scenario 6

Topic/Evaluation component	**Selected component (X)**	**Test scenario ID (see D5.04)**
API		
Camel model upload		
Initiate deployment process		
Get application status	X	
UI		
Web based UI for application view: Application view	X	
Web based UI for application view: Deployment view	X	
Eclipse based editor of the CAMEL: CAMEL Model validation		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Eclipse based editor of the CAMEL: Syntax completion		
BigData management		
Big data application deployment optimization		
Big data application deployment execution		
Big data application monitoring and reconfiguration	X	
Data locality awareness - features related to data locality and data movement.		

2.3.7 Evaluation Scenario 7: Template-based Utility Function Creation

Melodic offers several possibilities to maintain and extend the underlying model (including utility function). The evaluating person will evaluate both the metadata schema editor ('muse') and the (CAMEL) application model editor. Muse, for instance, could be used to specify the actual weights on partial utility functions/metrics while CAMEL editor will take these weights in order to complete the definition of the overall utility function (or to validate the content of the existing one). This scenario contains the following steps:

- Viewing the current model/s
- Changing the current model/s
- Updating the applied model/s

This scenario aims to evaluate rudimental functionality as described above, and advanced (graphical) modelling abilities of Melodic and the provided editors:

- Level of abstraction
- Behaviour with very small and/or very large models

Table 9 Evaluation components of scenario 7

Topic/Evaluation component	**Selected component (X)**	**Test scenario ID (see D5.04)**
API		
Camel model upload	X	
Initiate deployment process		
Get application status		
UI		
Web based UI for application view: Application view	X	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Web based UI for application view: Deployment view	X	
Eclipse based editor of the CAMEL: CAMEL Model validation	X	
Eclipse based editor of the CAMEL: Syntax completion	X	

2.3.8 Evaluation Scenario 8: Extended Stress Test

Such an extended stress test is supposed to allow an evaluation of Melodic, especially its functionalities and components under heavy load. Other than the scaling scenario, the focus lies on the platform itself.

Every functionality that potentially represents or handles a complex task should be considered for such an extended stress test.

Table 10 Evaluation components of scenario 8

Topic/Evaluation component	**Selected component (X)**	**Test scenario ID (see D5.04)**
Initial deployment		
Installation and deployment of a N-component application on M different Cloud Providers	X	1.3
Installation and deployment of a N-component application in Docker containers on M different Cloud Providers	X	1.6
Installation and deployment of a N-component application, where X component are installed in a Docker container and Y on a normal VM on M different Cloud Providers	X	1.7
Deployment requirement enforcement.	X	1.8
Installation and deployment of a N-component application on M different Cloud Providers with more advanced set of requirements, like non-functional ones.	X	1.9
Metric management		
Built-in raw metrics collection	X	2.1
Custom raw metrics collection	X	2.2
Composite metric collection	X	2.3
Event generation	X	2.4
Local reconfiguration		
Scale out application	X	3.1
Scale in application	X	3.2

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Global reconfiguration		
Attributes of used VM offerings changed	X	4.1
Global reconfiguration	X	4.2
Reasoning		
Linear constraints and optimization solving - CP Solver	X	5.1
Linear constraints and optimization solving - MILP Solver	X	5.2
Linear constraints and optimization solving - LA Solver	X	5.3
Non-linear constraints and optimization solving - CP Solver	X	5.4
Non-linear constraints and optimization solving - LA Solver	X	5.5
API		
Camel model upload		
Initiate deployment process		
Get application status	X	
Fault handling		
Temporary unavailability of Melodic platform components	X	6.1
Temporary unavailability of BPM - verifying proper system behaviour after BPM recovery.		6.2
Temporary unavailability of Cloud Provider		6.3
High Availability Component configuration	X	6.4
Performance		
Response time while solving complex allocation problems	X	6.5
Dynamic scalability within one Cloud - verification of the execution time		6.6
Dynamic scalability testing for multi-Cloud feature (using two different locations)		6.7
Counting Compute Resource Overhead of Melodic introduced over its host machine		6.8

2.3.9 Evaluation Scenario 9: Backup and Recovery of Melodic

Melodic platform installations shall support both backup and recovery with minimal administration and/or configuration overhead. Possible use cases are broken platforms or the necessity to move from one machine to another.
It is currently not decided whether Melodic will support backup and recovery directly or if such

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

actions have to be performed manually by the user on the underlying OS. The scenario contains the following steps:

- Configure Backup
- Perform Backup
- Store Backup data
- (Install Melodic)
- Stop the platform
- Load backup data
- Execute recovery
- Perform verification (e.g., with scripts as in Scenario 1)

Table 11 Evaluation components of scenario 9

Topic/Evaluation component	**Selected component (X)**	**Test scenario ID (see D5.04)**
Unified administration procedure		
Unified starting, stopping and restarting of Melodic platform	X	8.5
Configuring backup	X	8.6
Executing backup	X	8.7
Recover Melodic platform	X	8.8
Monitor Melodic platform		8.9

2.3.10 Evaluation Scenario 10: Monitor System Status

Melodic platform installations shall support the monitoring of the deployment as it is usually done in server/cloud environments by both developers and IT administrators. The system status contains:

- Deployment status
- Application and component specific status
- Platform status
- Underlying machine status
- Output of continuous logging information

Table 12 Evaluation components of scenario 10

Topic/Evaluation component	**Selected component (X)**	**Test scenario ID (see D5.04)**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Metric management		
Built-in raw metrics collection	X	2.1
Custom raw metrics collection	X	2.2
Composite metric collection	X	2.3
Event generation	X	2.4
API		
Camel model upload		
Initiate deployment process		
Get application status	X	
UI		
Web based UI for application view: Application view	X	
Web based UI for application view: Deployment view	X	
Eclipse based editor of the CAMEL: CAMEL Model validation		
Eclipse based editor of the CAMEL: Syntax completion		
Fault handling		
Temporary unavailability of Melodic platform components	X	6.1
Temporary unavailability of BPM - verifying proper system behaviour after BPM recovery.	X	6.2
Temporary unavailability of Cloud Provider	X	6.3
High Availability Component configuration	X	6.4
Performance		
Response time while solving complex allocation problems	X	6.5
Dynamic scalability within one Cloud - verification of the execution time	X	6.6
Dynamic scalability testing for multi-Cloud feature (using two different locations)	X	6.7
Counting Compute Resource Overhead of Melodic introduced over its host machine	X	6.8
Security		
Method invocation by programmatic access - Successful Authentication	X	7.1
Unsuccessful authentication	X	7.2

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Successful Authorisation Request	X	7.3
Unsuccessful authorisation request	X	7.4
Unsuccessful user authorisation with administrator privileges	X	7.5
Logging within Melodic platform	X	7.6
Unified administration procedure		
Unified starting, stopping and restarting of Melodic platform		8.5
Configuring backup		8.6
Executing backup		8.7
Recover Melodic platform		8.8
Monitor Melodic platform	X	8.9

2.3.11 Evaluation Scenario 11: Platform Security

Melodic platform installations shall satisfy the partner's individual requirements regarding platform's operational security. Therefore, a separate security evaluation scenario (decoupled from other evaluation scenarios) with the following details is planned:

- Rights are correctly handled and clearly separated between different user roles on both
  - UI
  - and API level
- SSL is used along all available web interfaces
- Authentication and authorisation happen in an appropriate way (per use case)
- Credentials are stored in a safe state-of-the-art way

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Table 13 Evaluation components of scenario 11

Topic/Evaluation component	Selected component (X)	Test scenario ID(see D5.04)
Fault handling		
Temporary unavailability of Melodic platform components		6.1
Temporary unavailability of BPM - verifying proper system behaviour after BPM recovery.		6.2
Temporary unavailability of Cloud Provider		6.3
High Availability Component configuration		6.4
Security		
Method invocation by programmatic access - Successful Authentication	X	7.1
Unsuccessful authentication	X	7.2
Successful Authorisation Request	X	7.3
Unsuccessful authorisation request	X	7.4
Unsuccessful user authorisation with administrator privileges	X	7.5
Logging within Melodic platform	X	7.6
User management		
Adding user		8.1
Removing user		8.2
Updating user password	X	8.3
Updating user profile		8.4
Unified administration procedure		
Unified starting, stopping and restarting of Melodic platform		8.5
Configuring backup		8.6
Executing backup		8.7
Recover Melodic platform		8.8
Monitor Melodic platform	X	8.9

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

2.4 Maximising Utility to Select the Best Application Configuration

The key information to run some of the evaluation scenarios described below is the reasoning process used to identify the best deployment solution. As a whole, Melodic should provide an application provider with a solution to deliver application and infrastructure services to a large numbers of final users taking into account various and even evolving requirements and Cloud Offers (COs). In the scope of WP6, we will consider the cases provided by the Melodic use case partners as presented in chapter 0. Each of such use case has its own requirements in terms of optimisation objectives and non-functional constraints. To meet the expectations of users in a cost-effective manner, the Melodic platform must support numerous deployment management decisions that satisfy diverse objectives, for example to meet Service Level Objectives (SLOs) while minimizing overall costs.

The selected infrastructure providers will themselves take over the infrastructure decisions like concrete final locations of the application and which host can be switched off. However, allocating virtual hosts to an application in the Cloud, in the context of unpredictable workloads, involves making deployment reconfiguration decisions for the use case partners, such as when and where to relocate an application on one or more Cloud offerings.

The deployment change decisions can be made with regards to meeting different objectives. Considering user satisfaction as a key objective would highly depend on the expectations of the user and cannot be easily captured and foreseen. Even if an agreed Service Level Agreement (SLA) is respected, this will not guarantee that the service performance level of the deployed application is leading to the full satisfaction of the user.

In the scope of the Melodic use cases, we will not consider end user satisfaction as a direct objective. The objectives and related metrics are provided by the platform users. Use cases deployment optimisation objectives could be globally related to four measurable overall property categories:

- service performance which involves objectives in terms of network latency, security, and response time (possibly limited to the minimum of a Service Level Objective - SLO)
- cost (e.g. infrastructure, operational, setup, migration, and SLO violation cost) (e.g., seeking to maximize the return on investment)
- computing power
- service reliability (including availability)

The Melodic use case providers do not expect support for the management of Service Level Agreement (SLA) as a whole. Anything related to SLA violation should be associated with an SLO metric that is then formalised in constraints and optimisation objectives.

A “utility-based approach” could be followed in the scope of Melodic to optimise the use case deployment. This kind of approach has already been applied to a range of applications, from

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

workflow scheduling on grids to data centre cooling. It could also be applied to flexible application deployment in the Cloud. A utility function could be thus defined specifying the overall goal of the deployment according to the user's expectations, and a deployment optimization algorithm explores alternative deployment solutions, to identify the one that maximizes the user utility as defined in the defined utility function. The focus in Melodic is on providing quite rich utility functions which incorporate multiple (SLO) metrics while they are also expressed in such a way that the decision making (i.e., search for an optimised solution) does not concern only the initial application deployment but also its reconfiguration.

As part of the overall evaluation framework, the following chapters and related annexes provide guidance and illustrations on an overall "utility based approach" that should be used for deployment configuration optimisation. This guideline will be crucial for the use case partners while leading their own Melodic platform implementation prior to final evaluation. The partner will test the implementation feasibility and relevance of this approach within the next WP6 steps. At the end, the evaluation results will highly depends on the adequate use of this approach for each use case.

2.4.1 Mastering Optimisation Complexity

An application is considered to consist of ***application component instances*** performing the application logic. Scalability of the application is ensured by *instantiating* one or more of the application types as needed, or more copies of one type. As an example, consider a simple web shop application consisting of a back end business logic server, a client database server, a webserver connecting the clients, and optionally a load balancer. Each of these is an *application component type* and at the exception of the load balancer, at least one instance of each component is necessary for the application to run. As more clients come in, more web servers are needed and with the second web server type instance, a first load balancer must be instantiated. As the shop grows further, more back end servers may also be needed. The utility of the *application* is therefore bound to its ability to serve the business' clients. When deployed in the Cloud, each application component will typically be deployed or instantiated in a ***virtual machine* (VM)**. One virtual machine may be capable of hosting multiple application components; thereby the mapping is not necessarily one-to-one between the set of application component instances and the set of virtual machines used to host the application.

It seems to be a common wish for most users of the Cloud to minimise cost and cost only. The cost in the Cloud is measured based on the used virtual machines, although in principle, the scaling of the application and the selection of the virtual machines to be used to host the application are orthogonal problems. Thus, to make a deployment decision one will need to do a two-step optimisation:

- Decide on the number of instances needed for each component type under the current

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

execution context; and

- Decide on the mapping of these instances to a set of available virtual machines so that the requirements of the instances grouped on a virtual machine is satisfied.

The first sub-problem may require various trade-offs between several aspects and the ***utility function*** is supposed to provide the application owner's view on a particular configuration's ***utility*** for its said purpose. The second part is normally only a *selection* process where the set of available virtual machines are filtered based on how each ***node candidate***, which is a generalised description of a virtual machine, satisfies the (combined) requirements of the application components that the node candidate is supposed to host. Then, the less costly instance of this filtered set will be the instantiated virtual machine executing the mapped components.

There are two reasons for this two-step approach:

- Separation of concerns since the two optimisation problems are orthogonal; and
- Scalability in finding the optimal *application configuration* that maximises the *application utility*.

For the second item, it should be noted that the application utility is the ability of the application itself to satisfy its requirements. This is independent from the virtual machines used to host the application, but it is generally not dependent on the Cloud provider of the machines and the location of the virtual machines. The scalability is an issue because the problems are generally *discrete* in that one has to choose a value of an *application component type attribute* from a discrete set of options. This implies that finding the maximal utility is a *combinatorial optimisation problem*. Algorithms for solving such problems are exponential in complexity. Consider an algorithm exponential with complexity $e^{a\,n}$ for n variables values to be chosen from. If one doubles the number of variable values, the time to find a solution increases by a factor of

$$(e^{2a\,n}/e^{a\,n})=e^{(2a\,n\,-\,n)}=e^{a\,n}$$

which is in general more than twice the time. It is therefore imperative to keep the number of variables as few as possible and the domain of the variables as small as possible for the problem to be tractable. The typical example is to decide on the number of instances for an *application component type,* which will be one integral variable assigned a numerical value, rather than deciding on the VM type taken from potentially a huge set of *node candidates* and leading to one variable for each instance of the *application component type*.

2.4.2 The Application Component Type

The application consists of a set of components, which are defined at the *type* level. Instances of the types are bound together to form the application logic, and the constraints relates the number of instances of one application component type to the other types. For instance, if there are more

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

than one web server to be deployed, *i.e.* the cardinality of the web server component type is larger than unity, then there should also be deployed a load balancer. The list of *application component types* will be built dynamically during design phase as the various components are defined.

```
ApplicationComponentTypes={};
```

2.4.3 The Requirement Attributes

Each *application component type* has certain ***requirement attributes***. These can be the number of cores needed by an instance of this application component, the amount of memory needed by this component, or the location needed by this component. It can also be the cloud provider needed, and the number of instances of the component needed in the application configuration. The set of attributes used in the examples of this notebook is

```
RequirementAttributes = {Cores, Memory, Provider, Instances};
```

By default, these are all set to undefined when a new application component type is defined, and if the type is already defined, nothing will be done.

```
SetUndefinedRequirements[ComponentID_Symbol] /; Not[ MemberQ[ ApplicationComponentTypes,
ComponentID ] ] := (
   Map[ (ComponentID[ # ] ^= Undefined )&, RequirementAttributes];
   AppendTo[ ApplicationComponentTypes, ComponentID ]
   );
SetUndefinedRequirements[ ComponentID_Symbol ]/; MemberQ[ ApplicationComponentTypes ,
ComponentID ]:= Null;
```

Each of the attributes can have either a single value, or be represented by a discrete set of values. Or be a range from a minimal value to a maximal value. In order to ensure that the attributes are properly set for a particular *application component type* several functions are provided. The *core* and *memory* attributes can be treated the same in that they can be either a number or an interval; in both cases zero is not a valid value.

```
SetRequirement[ TheAttribute_Symbol, ComponentID_Symbol, Value_Integer ] /; TrueQ[ MemberQ[{
Cores, Memory },TheAttribute] && Positive[ Value ] ]:= ( SetUndefinedRequirements[
ComponentID ]; ComponentID[ TheAttribute ]^= Value );
SetRequirement[ TheAttribute_Symbol, ComponentID_Symbol, Value_Interval ] /; TrueQ[ MemberQ[{
Cores, Memory },TheAttribute] && Positive[ Min[Value] ] && Positive[ Max[Value] ]] := (
SetUndefinedRequirements[ ComponentID ]; ComponentID[ TheAttribute ]^= Value );
```

The number of *Instances* of an *application component type* is a similar attribute; however, in this

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

case zero is an allowed value, and the acceptance test on the values will therefore be slightly different.

```
SetRequirement[ Instances, ComponentID_Symbol, Value_Integer ] /; NonNegative[ Value ]:= ( SetUndefinedRequirements[ ComponentID ]; ComponentID[ Instances ]^= Value );
SetRequirement[ Instances, ComponentID_Symbol, Value_Interval ] /; TrueQ[ NonNegative[ Min[Value] ] && Positive[ Max[Value] ]] := ( SetUndefinedRequirements[ ComponentID ]; ComponentID[ Instances ]^= Value );
```

The *provider* attribute is a non-empty set of possible Cloud providers for the instances of this component type. However, this only applies if the provider is given. It is possible to leave it undefined if any possible Cloud provider accessible to the application owner can be used.

```
SetRequirement[ Provider, ComponentID_Symbol, Value_List?VectorQ ] /; TrueQ[ Length[ Value] >0 ]:= ( SetUndefinedRequirements[ ComponentID ]; ComponentID[ Provider ]^= Value );
```

The vector of all the values of all requirement attributes for all *application component types* is called the ***application configuration***, and the *utility* is evaluated for this configuration.

2.4.4 Selecting the Node Candidates

The price to pay for splitting the optimisation problem in two separate parts is that the selection of node candidates has to be made based on the *application configuration*, which is the result of the first optimisation problem, and the selection has to be deterministic reflecting the application configuration. The most typical aspect of the VM that users want to include in the *utility* is the cost of the VM. However, since most users want to minimise cost, it makes sense to select the cheapest possible VM from the set of *node candidates* satisfying the requirements of an *application component type* when instantiating that type.
A *node candidate* is described by a set of attributes, where many of these attributes correspond to the attributes of the *application component types*. In addition, a node candidate will typically have assigned a price.

```
NodeAttributes = { Cores, Memory, Provider, Price };
```

The node candidate to host an instance of an *application component type* must provide enough resources for the instance of that *application component type*. However, there is nothing preventing the node candidate to over-provision resources. For instance, if an *application component type* needs 4 cores to run, then all node candidates with at least 4 cores will match this requirement. Note also that although the *requirement attributes* may be sets or ranges, the *node candidates'* attributes are always fixed values.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

The following functions implement the attribute match for the various variants, starting with the simplest case where a single numerical value is matched with a single numerical value.

```
AttributeMatchQ[ NodeValue_?NumericQ, RequirementValue_?NumericQ ]:= TrueQ[ RequirementValue
<= NodeValue ];
```

If the requirement attribute value is an *interval*, the match ensures if the node candidate attribute's value is in the interval given by the requirement attribute. Although this would in principle be correct, one must remember that the requirement is what an instance of the *application component type* needs to run, but in theory it would not harm to give the instance more resources, *i.e.* exceed the upper bound. Consider as an example the attribute memory: a component can require [10 GB, 15 GB] to run. It should then obviously not be allocated to a node candidate offering only 8 GB of RAM, but requiring strict conformance to the interval also it? excludes a node candidate offering 16 GB of RAM since this is outside the interval required. Thus, even if the last one GB of RAM will not be used if the application component runs on this node candidate, the node candidate is certainly able to host the application component and should not be excluded, also because it would be very strange to find a *node candidate* offering an amount of memory in the given interval. The comparison is therefore made against only the lower limit of the requirement interval.

```
AttributeMatchQ[ NodeValue_?NumericQ, RequirementValue_Interval] := TrueQ[
Min[RequirementValue] <= NodeValue ];
```

The last case to consider is when the requirement attribute is a list for which there must be an exact match for one of the members. The requirement attribute set will only match if it is a vector, i.e. a flat list of elements.

```
AttributeMatchQ[ NodeValue_, RequirementValue_List?VectorQ ] := MemberQ[ RequirementValue,
NodeValue ];
```

It could also be that an *application component type* attribute is undefined, which means that any *node candidate* attribute value should match this attribute

```
AttributeMatchQ[ NodeValue_, Undefined ]:= True;
```

It is obvious from the above definitions of the two sets of attribute types that the match can only be done on attributes that exist in both sets.

```
CommonAttributes = Intersection[ RequirementAttributes, NodeAttributes ];
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Selecting a node candidate for a component type instance is done by matching all set requirements for the common attributes, and applying the logical AND to the outcome of the tests for each of the attributes. A node candidate is basically a virtual machine if its entire attributes match in the sense described above.

```
NodeCandidateMatchQ[ NodeID_Symbol, ComponentID_Symbol ]:= Apply[ And,
   Map[ AttributeMatchQ[ NodeID[ # ], ComponentID[ # ] ]&, CommonAttributes ]
   ];
```

Finding out which *node candidates* can be used to host an instance of an application component is then just checking the node candidate match for each possible node candidate.

```
SelectNodeCandidates[ ComponentID_Symbol, PossibleNodeCandidates_List?VectorQ ]:= Select[
PossibleNodeCandidates, NodeCandidateMatchQ[ #, ComponentID ]& ];
```

This will yield a set of node candidates, but one would normally be interested in the cheapest node candidate. This requires a pricing model, which has been exemplified in the examples in Annex 2.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

3 The Definition Phase

The definition phase is the second phase of the Melodic Evaluation Framework, and concerns all activities that should be performed to formally define a measurement scheme.

This phase was already started with the definition of measurement goals, which were derived from the evaluation objects identified during the planning phase, both from the technical and business perspectives.

The following steps will be addressed later on in the scope of the upcoming WP6 deliverables. It will consist in:

- the definition of questions with respect to the measurement goals, to support data interpretation towards a measurement goal.
- the definition of metrics, which provide all the quantitative information to answer the questions and the verification of metrics for consistency and completeness.
- the production of a GQM plan that serves as a roadmap that contains indicative goals, questions, and metrics for executing the Evaluation Framework.

The following sections describe the preliminary results of the definition phase. However, these results are indicative as part of the overall framework and subject to further specific adjustments according to the evaluation scenarios.

3.1 Definition of Measurement Goals

The first step in the definition process is the definition of formal measurement goals. These validation objectives are derived from the evaluation objects and components, which are already identified in the preceding planning phase. Measurement goals have to be defined in an understandable way and with a clear structure.

The following table template, Table 14, underpins a generic evaluation goal's purposes based on the original GQM method. The Melodic measurement goals were defined accordingly both from the technical and the business perspective.

Table 14 GQM goal definition template

Analyse	*Clear evaluation object identification*
For the purpose of	*Understanding, controlling or improving the object?*
With respect to	*The particular object quality focus*
From the point of view	*The concerned evaluation group(s)*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

In the context of	*The environment in which the measurement takes place*

3.1.1 Technical Perspective

Well-known technical software evaluation metrics typically rely on the number of failures (also called bug or defect) that occur and the time required rectifying the failures. A detailed description of different software quality metrics and methods for applying the metrics is presented in [3]. The basic distinction made here is between software product quality, customer satisfaction and maintenance quality. Most of the metrics presented for product and maintenance quality are first relevant for evaluation from the developer perspective (mean time to failure, defect density, defect removal effectiveness, fix backlog and backlog management index, etc.). These metrics are pragmatically applicable in the scope of WP5 as there is a central issue tracking the system in place, which can be used for analysing in the long run the failures that occur and the time to solve them.

However, in the frame of the Melodic technical evaluation that should be led in WP6, we are mainly targeting evaluation of the components from the user perspective (where the user can be a developer as well as a deployment administrator but anyway someone consuming the Melodic platform to deploy a software and not the developer of the Melodic components themselves). In the scope of this work-package, we will therefore rely on questionnaires and standard monitoring information to evaluate the overall platform quality.

Following this approach, the measurement goals of Melodic are developed based on the ISO 25010 Software product Quality [4], even if such a conventional framework does not effectively support Melodic-specific quality aspects and take into account project-specific priorities. The ISO model offers the core on top of which the priorities can be expressed.

Figure 4 ISO 25010 Software Product Quality Characteristic [4]

As agreed between the use case partners based on the outcome of questionnaires, in the scope of the Melodic project, we will consider the following key quality parameters while leading component technical evaluations:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

1. **Functional suitability**: satisfaction of the stated or implied needs. Measures to be considered may include functional completeness, correctness and appropriateness.
2. **Reliability**: the capability of the Melodic platform to maintain its level of performance under specific conditions. Measures to be considered may include maturity, fault tolerance, recoverability and availability.
3. **Usability**: the effort needed for using the platform. Measures to be considered may include understandability, learnability, operability and attractiveness.
4. **Performance**: the level of performance exhibited of the platform. Measures to be considered may include time behaviour and resource utilisation.

Following is an overview of the selected quality focus including a first attempt of breakdown to concrete attributes and metrics. These attributes and metrics will be identified in details in a later stage in WP6.

Figure 5 Quality focus, breakdown to concrete attributes and metrics samples

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

According to the focus on these quality attribute categories, we will apply the GQM goal definition approach to the evaluation components.

This is actually a pattern which will be instantiated according to all the relevant evaluation components and evaluation groups. This leads to the following measurement goals according to the evaluation component:

" Analyse [EVALUATION COMPONENT]

for the purpose of controlling and improving the objects with respect to

1. *Functional suitability*
2. *Reliability*
3. *Usability*
4. *Performance"*

from the viewpoint of [ADMINISTRATOR OR DEVELOPERS]

in the context of [EVALUATION SCENARIO]"

3.1.2 Business Perspective

The final performance evaluation to be performed is the business impact of Melodic. The use case partners should evaluate the overall business performance of the integrated platform by referring to the performance of a similar system employing their current choice in technologies.

The focus will be, for instance, on comparing the delay of typical actions (i.e., selection of deployment configuration, deployment operation, etc.). The business evaluation in the integrated case should also consider and refer to the standalone use case scenario processes by considering both comparable and dissimilar contexts (i.e., by having the various relevant processes running both on the Melodic system and on a current solution).

From a business point of view, many different goals need to be addressed at different levels. They can be all integrated into an overall competitive priorities framework but the detailed evaluation objects will be depending on the individual evaluation cases themselves.

Both SaaS Business Managers and SaaS Deployment administrator profiles have to be considered.

3.1.2.1 Business Priorities

Following is an overview of the Melodic competitive priorities identified by the consortium. They will constitute the overall measurement goals from a business perspective.

1. **Speed:** Computation, speed of technical deployment set-up, update or training

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

2. **Cost reduction:** Every organization wants to keep the following kinds of costs low. 1) Set-up cost: personnel, infrastructure and training cost; 2) Operational cost: infrastructure operation (e.g., lower cost of cloud), maintenance and personnel cost
3. **Quality:** for end user: latency; for administrator: increased data confidentiality and reallocation transparency and improved and predictable application performance for end users
4. **Reliability:** Reliability is the ability to provide continuous deployment and reconfiguration services that can defensibly be trusted within a time-period. This may also encompass mechanisms designed to increase and maintain the reliability of the Melodic platform. In the end, from a business perspective, reliability overrides all other factors. It does not matter how cheap and fast the deployment is: if the final user cannot trust that he will get the hosted application in time, at the right quality, he will be lost. Reliability inside the provider organization is also very important as it saves time and money and gives stability within the organization.
5. **Flexibility:** This priority is related to the ability to being able to change what, how and when so that the evaluator is able to evaluate four types of requirements:
    - Application flexibility - ability to introduce new or modified applications
    - Cloud flexibility - ability to manage a wide or mix variety of Cloud Offers
    - Deployment volume flexibility - able to change the level of deployment output according to needs
    - Utility flexibility - ability to change SaaS delivery model

The flexibility of the Melodic platform is also important for the platform user as it speeds up responses to change, saves time and maintains reliability.

As a whole, a trade-off have to be considered by a Melodic provider. A provider may "sacrifice" one priority to improve another performance objective. The Melodic competitive priorities have both external and internal effects that are the interest of business managers and deployment administrators, respectively.

3.1.2.2 Business Measurement Goals

The business priorities identified above lead to the following measurement goals according to the use case scenario and to the evaluation group (administrator or business manager):

" Analyse [USE CASE SCENARIO]

for the purpose of controlling and improving the objects with respect to

1. *Speed*
2. *Cost*
3. *Reliability*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4. *Flexibility*
5. *Quality*

from the point of view of [ADMINISTRATOR OR BUSINESS MANAGER]
in the context of [USE CASE]"

Nevertheless, the quality focuses cannot be fully implemented through the business priorities and are further defined for each use case (see Chapter 4).

3.2 Definition and Review of Questions

With respect to the measurement goals, questions have to be defined to support data interpretation towards a measurement goal. As goals are defined at an abstract level, questions act as refinements of goals to a more operational level, which is more suitable for interpretation. By answering the questions, one should be able to conclude whether a goal is reached or not.

The questions of the Evaluation Framework will be the basic translation from goals to metrics. Therefore, the questions will take a central role, not only during definition, but also during interpretation. Hence, it will be important to make sure that these questions are correct.

3.3 Definition of Metrics

Once the goals have been refined into a list of questions, metrics will be defined in order to provide criteria for all the quantitative information to answer the questions in a satisfactory way. Metrics or criteria are a refinement of questions into a quantitative process and/or product measurements. Furthermore, factors that could possibly influence the outcome of the metrics have to be identified. Factors that directly influence metrics also influence the answers to the questions to which the metrics are related. If the influencing factors are not to be considered during the definition of a measurement scheme, some conclusions or interpretations of the collected data may not be correct.

The goals, questions, and metrics of the Evaluation Framework have to be consistent and complete with respect to models of the validation objects. To safeguard this, consistency and completeness checks will be performed throughout the entire definition phase in project meetings. This action will be done in order to prevent missing definitions, incomplete, or inconsistent definitions. It is expected that definitions and metrics will be adjusted to comply with the respective goals, questions, and metrics definitions.

The mapping from goals to questions and also the mapping from questions to metrics will be achieved for each evaluation scenario in the scope of the upcoming WP6 deliverable (D6.2 and D6.6).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

3.4 Timetable for Executing the Evaluation Framework

Below in Table 15 is the interim version of the evaluation preparation and execution timetable, which has been designed, based on platform release availability. Despite the fact that each evaluation group and support is unique in its configuration, objectives, validation perspectives, validation groups, and validation definition, it is vital to define a synchronized timeframe for the execution of the validation as the Melodic partners will be able to exchange experiences and lessons learned, further contributing to the continuous improvement of the platform and each demonstration.

The timetable below represents a rough-grained schedule for the evaluation preparation and execution and allows each partner to plan more fine-grained activities according to the specific requirements of the evaluation groups and setup. The implementation activities allow deploying the evaluation scenarios while the platform releases are becoming available. The evaluation execution is planned to take place between M28 and M36.

Table 15 Time table for executing the evaluation framework

Month	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
Release	**R1.5**									**R2**				**R2.5**								**R3**
Use case plan																						
1 - Setup Melodic																						
2 - Add Application																						
3 - Deploy Application																						
4 - Optimization																						
5 – Local reconfiguration																						
6 - Undeploy Application																						
7 - Template-based UF creation																						
8 - Extended Stress Test																						
9 - Backup and Recovery																						
10 - Monitor system status																						
11 - Platform Security																						

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Targeted Deliverable	**1st feedback (D6.2)**	**2nd feedback (D6.3)**	**Data collection and interpretation (D6.4)**	**Results (D6.5)**

The definition and review of questions as well as the definition of metrics will take place according to the deployment plan of the evaluation scenarios. D6.2 will include the outcomes for evaluation scenarios 1, 2, 3 and 4. The next deliverable D6.3 will address the other evaluation scenarios. In order to benefits from the latest development outcome from WP5 while starting evaluation execution, an additional intermediate release (R2.5) is planned for month 28.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4 The Melodic Use Cases

Following the definition of the performance evaluation to be applied in order to show the usability and sustainability of the developed Melodic platform, this chapter is dedicated to the definition of specific business scenarios. Covering the use cases and functional requirements collected until and analysed in the previous deliverables, the business scenarios aim at demonstrating the important functionalities and their benefits to European Cloud providers and Software SMEs.

Altogether four business demonstrations reflect the main use case topics as defined above. Therefore, the following four business demonstrations are defined in order to demonstrate the usability and sustainability of the Melodic platform.

In Melodic, we have carefully selected four use cases, introduced in D2.1, each representing a separate class of problems addressed by Melodic as well as demonstrating different models of commercial exploitation.

The CET use case will demonstrate how Melodic will assist CET in providing close to real time processing of geo-dispersed big data on vehicle mobility for advanced, on-demand modelling of traffic in cities. Melodic will not only make this innovation technically possible, but also enable CET to provide such a service to many cities for an acceptable price, both under permanent contract as well as on demand, e.g., when a large event is planned in the city, or for crisis management purposes. With the second application, CET will demonstrate how the highly sensitive data on people mobility, acquired from and stored in different locations, including private clouds of telecom operators, can be securely accessed for on-demand processing and delivering affordable data analysis services to SMEs from different sectors, such as e-commerce, retail or tourism, potentially unlocking innovative data-driven business models in these sectors.

7Bulls, as experienced enterprise-class software system provider, will demonstrate how Melodic can speed up and simplify the development of big-data applications with two use cases: (1) a specialised application for processing genome data for medical purposes that imposes strict data confidentiality requirements. It will benefit from its migration to Melodic through the reduction of the infrastructure cost for its execution as well as an improvement of its execution time, (2) a Melodic-powered multi-cloud value-added service for a cloud operator. Under both use cases, 7Bulls will exercise and validate a typical OSS-based exploitation model, where a software company delivers a solution and supports based on OSS resources. Noteworthy, however, the latter use case also represents a very interesting exploitation model for Melodic. The platform can be integrated with the offering of a cloud provider or a cloud broker platform, enabling a value added service.

CAS in turn will show how a large CRM software provider can integrate Melodic into an internal, dedicated application store and deployment platform as an innovative way to manage the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

extension and personalisation of the core system. Melodic will provide support for deployment of data-intensive extensions of the CAS CRM.

All these various use cases have a clear potential for virtually unlimited replication and adaptation, saving time and money for other data intensive application creators, providers and users, as well as paving the way for innovative services and business models in cloud and big data industries. The described use cases reflect the planning until February 2018 (M15).

4.1 Use case 1: A marketplace for data-intensive apps supporting deployment in multi-cloud

CAS Software AG is a provider of relationship management products. The strategy of CAS ranges from CRM to xRM (anything Relationship Management), an all-round approach for managing relations with all kinds of stakeholders: customers, employees, suppliers, partners, etc. The next step of this strategy is to propose the so called "app-in-app" concept by opening an app-store-like ecosystem of business apps for project management, work scheduling, contact management, etc. External developers will be allowed to add their own solutions/apps complementing the CAS CRM/xRM offering. This ecosystem will provide companies with access to a large selection of cross-platform web apps and will support data exchange and integration across different apps driven by the end-user needs. In this way, they can adapt the xRM solutions to the needs of their employees, tailor the functionalities they actually use on a daily basis, and select the best apps from different providers without storing all of them on their local computers. The xRM system operating in this environment will be fed with the data coming from the apps offering new opportunities in business relation management.

In the future, CAS would like to rely on multiple cloud providers while deploying apps purchased in their marketplace. The main challenges related to building this ecosystem are high costs of data storage and processing as well as data security. Typically, companies manage relationships with different types of stakeholders where the volume of data and security requirements may vary significantly. Some customer data, like price lists, are treated with more confidentiality than other. In addition, some customers are more sensitive about the security of their data in general.

Many companies cooperate from multiple locations in virtual teams, teleworking, telecommuting; yet, they still want to benefit from cloud technology to be able to securely share, process and synchronize data, in some cases in big volumes, between different physical, remote and mobile teams.

Importantly, such applications typically require high scalability of computation and data storage capacities depending on the number of users per each app and the amount of data it generates. Although any commercial cloud provider offers virtually unlimited scalability, the cost of vendor lock-in may turn out prohibitive. However, if they could have a tighter control over their data in

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

terms of security and confidentiality, access control and location transparency, they would definitely consider a public multi-cloud for handling peaks in computing and storage demands.

For performance and efficiency as well as demand-driven, managed (controlled) scalability and portability to multi-cloud, CAS would turn to the Melodic framework to help it manage the deployment of applications purchased in the marketplace according to the user-defined requirements. The app store would hold complete information about each app as well as the resource and security related requirements for deploying it. Then, the Melodic framework would use this information to identify the data intensity and sensitivity of specific apps and automatically assign them for supporting the respective application deployment across heterogeneous multiple cloud infrastructures, including private, and potentially public providers (for 3rd party apps).

Also, Melodic will be able to identify which components should be hosted together with the basic CRM/xRM system and which could operate from different locations without affecting their performance.

For CAS, the main benefit would be to efficiently manage the resources and control the costs related to running this ecosystem, by hosting different components or apps in different locations depending on the required security levels and data access constraints.

4.1.1 Overview

4.1.1.1 Technical Architecture

SmartDesign and the CRM backend server OPEN are Java-based and need a Java runtime environment. Additional apps can be either native apps, which are executed as part of the SmartDesign process, or HTML apps that are effectively embedded via iFrames[1] into the UI. In the latter case, a JavaScript library can be used to access the CRM's data. This way, even comparably complex apps, such as third party calendars, can be developed without the necessity of executing any additional Java code.

In case of the PicassoSearch, a self-contained Java JAR file is executed on a host that is known to the OPEN backend server. Technically, the OPEN server regularly updates the PicassoSearch with the deltas between actual and previous changes. The search is available and triggered within SmartDesign by a business operation (BO). Figure 6 depicts the overall architecture and component relations.

[1] https://www.w3schools.com/tags/tag_iframe.asp

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

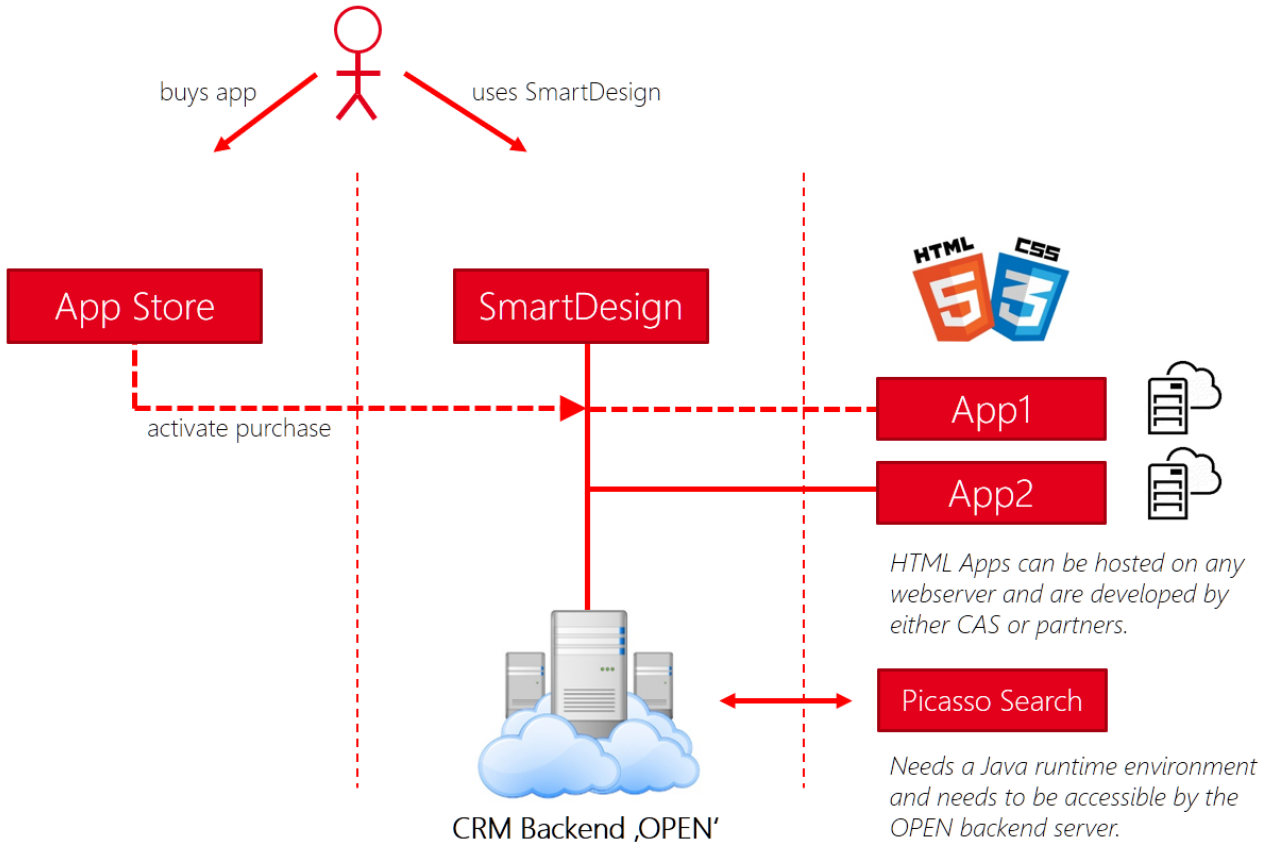


Figure 6 SmartDesign and App Store Architecture

4.1.1.2 Overall Business Environment

Table 16 Case 1 - Business environment

Business role		**Partner involved**
Deployed application user	Who is the final user of the deployed application?	*CRM platform customers*
Application provider	Who is providing the application to be deployed?	*CAS/third party developer*
Cloud provider	Who is providing the VMs?	*CAS*
Melodic platform user	Who is starting deployment execution and provide the model?	*Initiated by the CRM platform customer/partner but via our platform*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Melodic platform administrator	Who is administrating the platform?	*CAS*

4.1.1.3 Expected Benefits

Deployed application user: *SmartWe customer*

Table 17 Case 1 - Expected benefits for SmartWe customer

Benefit type	Benefits description
Speed	• Increased UI performance (user experience) • Increased processing time (e.g., for data import and export)
Cost	• Possibly lower product cost
Reliability	• Stable user sessions while using the product
Flexibility	• Resources can be allocated and also removed on-demand • Reaction to increasing or decreasing user numbers
Quality	• Increased overall experience due to perfectly balanced application servers • No delays, fast initialization times, fast data loading

Application provider: *CAS Software AG*

Table 18 Case 1 - Application provider expected benefits

Benefit type	Benefits description
Speed	• Serving main product (SmartWe) and Extensions, Apps at appropriate performance
Cost	• Reduced operation and hosting cost • Reduced personnel cost
Reliability	• Main product SmartWe has high availability
Flexibility	• New Apps can be added during runtime without stressing existing resources
Quality	• Partners and customers experience extensible, performant and highly available product
Other type of benefits	• Reaching more (potential) customers with our products (Apps) • Being able to compete on the mobile/cloud XRM market

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Cloud provider: *CAS Software AG*

Table 19 Case 1 - Cloud provider expected benefits

Benefit type	Benefits description
Speed	• Fast provisioning of resources
Cost	• Machines are used efficiently
Reliability	• Machines always host an available and accessible product • Machines have high availability leading to high overall quality
Flexibility	• New instances can be added easily when needed
Quality	• Resources are dynamically handled for a 'clean' operational state
Other type of benefits	• Being able to compete with other products that already employ public cloud solutions in the future • Opening the possibility to employ public cloud offers (at least additionally) to our current "self-hosting" approach

Melodic platform user: *CAS Software AG*

Table 20 Case 1 - Platform user expected benefits

Benefit type	Benefits description
Speed	• Fast and responsive UI -> efficient use • Fast executions after triggering an event -> fast results
Cost	• Low cost due to short training periods due to easy platform usage
Reliability	• Platform has high availability
Flexibility	• Platform can be adjusted and extended to the user's need • Different user roles are supported
Quality	• Users experience an easy to use system and can apply common patterns of using the platform • Responsible users are happy to work with the platform

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Melodic platform administrator: *CAS Software AG*

Table 21 Case 1 - Platform administrator expected benefits

Benefit type	Benefits description
Speed	• Fast reaction to new situations (e.g., new users, new apps) • Melodic related administration tasks can be done efficiently -> more time to spend for other tasks
Cost	• Efficient due to shorter platform usage (=administration) times
Reliability	• Platform has high availability, reduce time spent on ensuring availability to a minimum
Flexibility	• Grows with organizational and technical needs • Extensible according to administrator's skills
Quality	• Administrator can work according to his skill level (e.g., scripting, raw configuration editing etc.) • Acceptance along employees (users)
Other type of benefits	• Integrability: platform can easily be integrated (UI and technology-wise) into existing environments

4.1.2 Melodic Individual User Roles

Table 22 Case 1 - Individual user roles

Melodic generic role	Use case specific role name (i.e. the name that the partner will use)[2]	Description of the role in the use case (task, working environment, ...)
System administrator (responsible for the initial installation of Melodic)	***Melodic Administrator,*** *Administrator, SmartWe Developer*	*Will install and setup Melodic on the beta and test systems first. Usually has other administrative responsibilities.*
Application model provider	***Melodic Developer,***	*Knows own application and is able to describe and provide such a*

[2] Bold one (if present) indicates the most generic and probable one

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

(providing application data into CAMEL)	*Application Developer, SmartWe Developer, CAMEL Developer*	*CAMEL description (either handwritten or created with visual assistance). Can be provided by an external Application developer or by an internal SmartWe developer. External application providers create a simple CAMEL model based on a CAS specific template.*
Deployment rules provider (utility function and overall deployment constraints)	***Melodic Developer,*** *Application Developer, SmartWe Developer, Deployment Manager, Ecosystem Manager*	*Knows most relevant aspects about own application and is therefore able to abstract from its very individual necessities. Can either be done by an external application developer that contributes an App via the AppStore or by an internal SmartWe Developer and/or specialized IT staff.*
Dataset provider (providing cloud offers)	***Melodic Developer***	*Since our UCs focus on the private cloud/static cloud offer approach, this task is realistically done by someone that knows CAS deployment infrastructure very well.*
Melodic end-user (running operational deployment)	***Melodic Admin,*** *SmartWe/App Store/ Administrator*	*Will trigger deployments. Can be either a natural person or a SYSTEM user. In case of the App Store, a SYSTEM user would be the case.*
Application end-user (using the deployed application)	*SmartWe Customer*	*Uses the applications that were provided for him. Ideally, the existence of Melodic is completely unknown to him.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4.1.3 Evaluation Groups

Table 23 Case 1 - Preliminary list of evaluation group members

Last name	First name	Profile (dev., adm. or BM)	Company Unit	Specific role(s) in the scenario
Schork	Sebastian	dev, adm, BM	DCS	Mainly technical, might also contribute to BM
Schwichtenberg	Antonia	adm, BM	DCS	knows product development and product management
Bauer	Markus	BM	DCS	Head of DCS
Vuong	Julia	dev, adm, BM	DCS	Non-affiliated PM with experience in all areas
Terhorst	Jens	Dev	DCS	SmartWe and App Developer
Ristau	Dominik	Dev	DCS	SmartWe and App Developer
Erdei	Attila	Dev, adm.	SQS	Tester

4.1.4 Applications to be deployed

SmartDesign *(basis application whether standalone or with additional apps)*

It is the base technology of CAS Software AG's cloud solutions. SmartDesign provides a modern web-based CRM UI that can be employed together with both the company's cloud-based CRM backend 'OPEN' (resulting in **'SmartWe'**) and the long-standing on premise CRM solution 'Genesis World'. SmartDesign is both a product and a technology. By using its own Domain Specific Language (DSL) further apps within the CRM are supported, allowing customers to extend the platform to their individual needs. This is where the 'app store' UC connects by transferring SD Apps into the SmartDesign Client. SmartDesign is available as a web app as well as for mobile devices.

ContextService UI *(as an application representative within the app store scenario)*

The ContextService UI is part of the company's context information infrastructure that allows mobile and web clients to store contextual information to be later used for smart features. The UI

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

provides access to this information and is shipped as a comparably small-sized spring boot[3] based JAR file. It has been used within early releases as an Application that can be added (and removed) to an existing SmartDesign installation.

PicassoSearch/SmartSearch *(as a big data application)*

The PicassoSearch represents a data intensive application that can be added and removed to an individual SmartDesign installation. The Application can optionally be added to a customer's CRM platform which makes it suitable for being handled as an application that can also be used within the 'app store' UC.

The data-intensive character of the application is described in detail below.

4.1.4.1 Data Intensive Aspects

The big data aspect of the CAS UC is subject to releases 2.0 and higher. Here, the PicassoSearch[4] as a data intensive application is used. PicassoSearch allows users to perform smart searches over a large document store featured by elasticsearch[5]. The capacity of such a store can be up to 10 GB or more. Such a search engine itself works on data previously (and continuously) synced from the company's CRM backend.

4.1.4.2 Structural Application Model

Table 24 Case 1 - Structural application model

Criteria	**Value**
SmartDesign.	
RAM	*>= 6 GB*
CPU	*>= 2 Cores*
OS	*Debian Linux*
Storage	*20 GB*
Environment	*Java 8*

[3] https://projects.spring.io/spring-boot/
[4] https://www.cas.de/nc/de/presse/pressemitteilungen/details/article/erstes-crm-mit-integriertem-fan-prinzip-aufbruch-in-eine-neue-we-welt.html
[5] https://www.elastic.co/de/products/elasticsearch

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

ContextService UI	
RAM	*>= 2 GB*
CPU	*>= 2 Cores*
OS	*Debian Linux*
Storage	*10 GB*
Environment	*Java 8*
PicassoSearch/SmartSearch	
RAM	*>= 8 GB*
CPU	*>= 4 Cores*
OS	*Debian Linux*
Storage	*40 GB*
Environment	*Java 8*

4.2 Use case 2a: Data-intensive application for people flow (mobility) monitoring and analysis based on anonymised signalling data from mobile operator network

4.2.1 Overview

CET is a provider of traffic and mobility information services for private businesses and the public sector. Data about people flows is essential for governmental agencies, cities, municipalities, as well as private business owners. CET is using anonymised signalling data from mobile operator network for monitoring and analysis of people presence and travels. It includes analysis such as counting visitors of selected sites, advanced tourism statistics, origin-destination analysis as well as concentration measurement of people in real-time.

CET built its mobility tools in Python for its ease of use and availability of various, powerful data analyses capabilities. These applications however do not support distributed parallel processing and are limited to single machine. This is serious limitation when it comes to complex, large scale

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

spatial-temporal analysis. To overcome this issue CET will develop cloud-ready, scalable application for mobility analysis which will use a selected Big Data framework for efficient, large scale distributed computing.

Thanks to such approach CET will be able to use Melodic for optimised deployment and scaling of the application according to submitted analysis tasks and actual load.

4.2.1.1 Technical Architecture

The application architecture will be based on the big data platforms, such as Hadoop and Spark. CET will utilise all its experience and knowledge gained over the years of working with mobility data and implement improved algorithms into this new application. It will be able to take advantage of the parallel processing capabilities as well as of the Melodic optimised deployment, scaling and resource management. The application will use source data stored in a Hive/HDFS file system.

It shall be noted that this is a preliminary planned architecture and functionality of the system which is subject to adjustments during development process. CET will gradually release applications and their features.

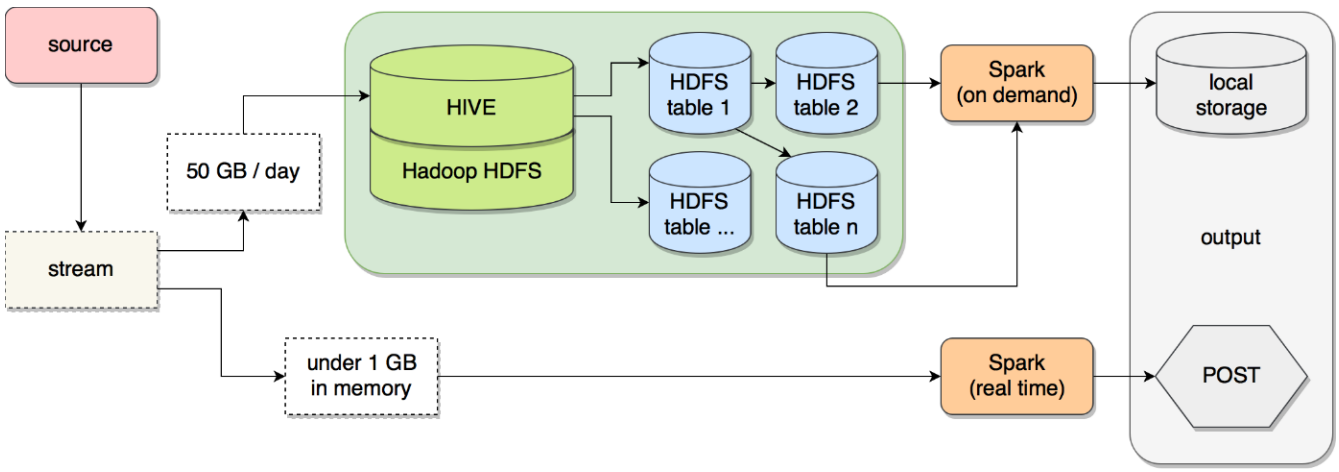


Figure 7 People flow monitoring application architecture with Melodic

4.2.1.2 Overall Business Environment

Table 25 Case 2 - Business environment

Business role		**Partner involved**
Deployed application user	Who is the final user of the deployed application?	CET is the only user of the application however output data is made available for the CET customers
Application provider	Who is providing the application to be deployed?	CET
Cloud provider	Who is providing the VMs?	Mobile operator or other cloud provider approved by mobile operator
Melodic platform user	Who is starting deployment execution and provide the model?	CET
Melodic platform administrator	Who is administrating the platform?	CET

4.2.1.3 Expected Benefits

Deployed application user: CET

Table 26 Case 1 - Expected benefits for deployed application users

Benefit type	**Benefits description**
Speed	• Controlled computing time of selected tasks. • Manageable performance of the application. • Faster reaction and adaptation.
Cost	• Decreased cost of testing and deploying of the application.
Reliability	• Stable performance of the system. • Increased availability.
Flexibility	• Resources are added or being released according to the real load.
Quality	• High overall quality of the system and implementation environment.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Application provider: CET

Table 27 Case 1 - Application provider expected benefits

Benefit type	Benefits description
Speed	• Shorter time of testing and deploying of the application.
Cost	• Decreased cost of testing and deploying of the application.
Reliability	• Highly available solution.
Flexibility	• Resources are added or being released according to the real load.
Quality	• High overall quality of the system and implementation environment.

Cloud provider: Mobile operator or private cloud provider

Table 28 Case 2 - Cloud provider expected benefits

Benefit type	Benefits description
Cost	• Decreased cost of the inhouse resources.
Flexibility	• Avoiding unnecessary use of resources. Resources are added or being released according to the real load.
Reliability	• Highly available solution.

Melodic platform user: CET

Table 29 Case 2 - Platform user expected benefits

Benefit type	Benefits description
Speed	• Shorter time of testing and deploying of the application.
Cost	• Decreased cost of testing and deploying of the application.
Reliability	• Highly available solution.
Flexibility	• Resources are added or being released according to the real load.
Quality	• High overall quality of the system and implementation environment.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Melodic platform administrator: CET

Table 30 Case 2 - Platform administrator expected benefits

Benefit type	Benefits description
Speed	• Finding problems faster thanks to platform monitoring.
Cost	• Reduced administration cost thanks to platform monitoring.
Reliability	• Reliability ensured by Melodic may simplify administration.
Flexibility	• Unified way of deploying various applications.
Quality	• Platform monitoring and other Melodic's tools may improve the quality of administration.

4.2.2 Melodic Individual User Roles

Table 31 Case 2 - Individual user roles

Melodic generic role	Use case specific role name	Description of the role in the use case (task, working environment, ...)
System administrator (responsible for the initial installation of Melodic)	Administrator	Person responsible for: • initial installation of Melodic • monitoring and maintenance of Melodic • solving technical issues with Melodic
Application model provider (providing application data into CAMEL)	Developer/ CAMEL developer	Knows application very well and is responsible for describing application model and providing application data in CAMEL
Deployment rules provider (utility function and overall deployment constraints)	Developer/ CAMEL developer	Knows application as well as business context including optimisation goals and is responsible for providing deployment rules in CAMEL.
Dataset provider (providing cloud offers)	Administrator/ CAMEL developer	Person responsible for providing cloud offers rules in CAMEL.
Melodic end-user (running operational deployment)	Administrator	Person responsible for: • installation and launch of the application • monitoring the application and solving technical issues

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Application end-user (using the deployed application)	Developer / Product owner	Person responsible for the delivery of the product.

4.2.3 Evaluation Groups

Following is a preliminary list of evaluators.

Table 32 Case 2 - Preliminary list of evaluation group members

Last name	First name	Profile	Company Unit	Specific role(s) in the scenario
Przeździęk	Tomasz	BM	CE-Traffic	Business and product manager, evaluating business impact
Novobilský	Jiří	BM	CE-Traffic	Business and product manager, evaluating business impact
Masata	Hynek	BM/dev/adm	CE-Traffic	Business and product manager, designing and developing application
Ficek	Michal	dev	CE-Traffic	Designing and developing application
Vlčinský	Jan	dev/adm.	CE-Traffic	Real-time systems admin, application developer
Jarmuż	Dominika	data/dev/adm	CE-Traffic	Mobility data specialist, preparation of datasets, testing and evaluating application
Tatar	Kinga	data/dev	CE-Traffic	Mobility data specialist, preparation of datasets, testing and evaluating application
Stec	Katarzyna	data/dev	CE-Traffic	Junior mobility data specialist

4.2.4 Applications to be deployed

CET will develop Advanced OD matrix analysis application able to calculate country wide detailed and accurate trip statistics. The application will use signalling data retrieved from mobile operator and stored in Hive/HDFS file system. The application will be based on the Big Data

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

framework to make use of the distributed parallel computing for scalability and high performance.

4.2.4.1 Data Intensive Aspects

Calculating mobility statistics is a data-intensive task. Signalling data coming from the mobile operator network as a continuous real-time data stream will be incrementing HDFS warehouse. Daily amount of retrieved data is about 50 GB. Depending on the project, it may require processing data from a single day, many weeks or even several months which means processing from 50GB to several TB of source data. In case of advanced statistics such as origin-destination matrices the task is even more demanding because of use computationally intensive clustering algorithms used to improve the accuracy of the results.

4.2.4.2 Structural Application Model

Table 33 Case 2 - Structural application model

Criteria	**Value**
RAM	>= 8 GB
CPU	>= 2 Cores
OS	Ubuntu

4.3 Use case 2b: Real-time traffic management based on the Floating Car Data and advanced traffic simulations

4.3.1 Overview

Road traffic has an impact on the whole society, economy and environment, so it is crucial to monitor and analyse it, predict its evolution and manage it properly. However, it is not an easy task, as road traffic is a complex phenomenon involving many heterogeneous agents (e.g. people, vehicles, public transport etc.) and depending on many factors (e.g. time of the day, day of the week, road closures, road works, incidents, weather, mass events etc.). It is also subject to traffic management policies and operational strategies (e.g. traffic signal settings).

To support monitoring, analysis, prediction and management tasks, it is necessary to have access to a real-world real-time and historical traffic data. It is also important to be able to perform what-if analysis (e.g. what happens when road is closed or accident has happened).

In this use case CET will use floating car data (FCD), machine learning and traffic simulations to address above mentioned challenges and demonstrate how Melodic platform can facilitate these data- and compute-intensive tasks.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4.3.1.1 Technical Architecture

The development will be based on the following existing applications and data resources:

- Floating car data system, which provides real-time traffic information about speed, travel-time, delay and level-of-service on the monitored road network.
- Archived traffic information.
- Traffic Simulation Framework (TSF) - application for running traffic simulations in microscopic and mesoscopic models in a large scale.
- TensorTraffic - TensorFlow-based tool for approximating outcomes of traffic simulations using neural networks. It uses as an input data produced by TSF.
- Traffic optimization module - application for searching through a large space of possible traffic signal settings and finding (sub)optimal settings using genetic algorithms.

The intended traffic management platform will be based on existing components, but it will also contain new modules for:

- calculating typical traffic profiles based on historical data
- calibrating TSF using typical traffic profiles
- short-term traffic prediction

It shall be noted that this is a preliminary planned architecture and functionality of the system which is subject to adjustments during development process. CET will gradually release applications and their features and utilise them to evaluate early versions of Melodic platform.

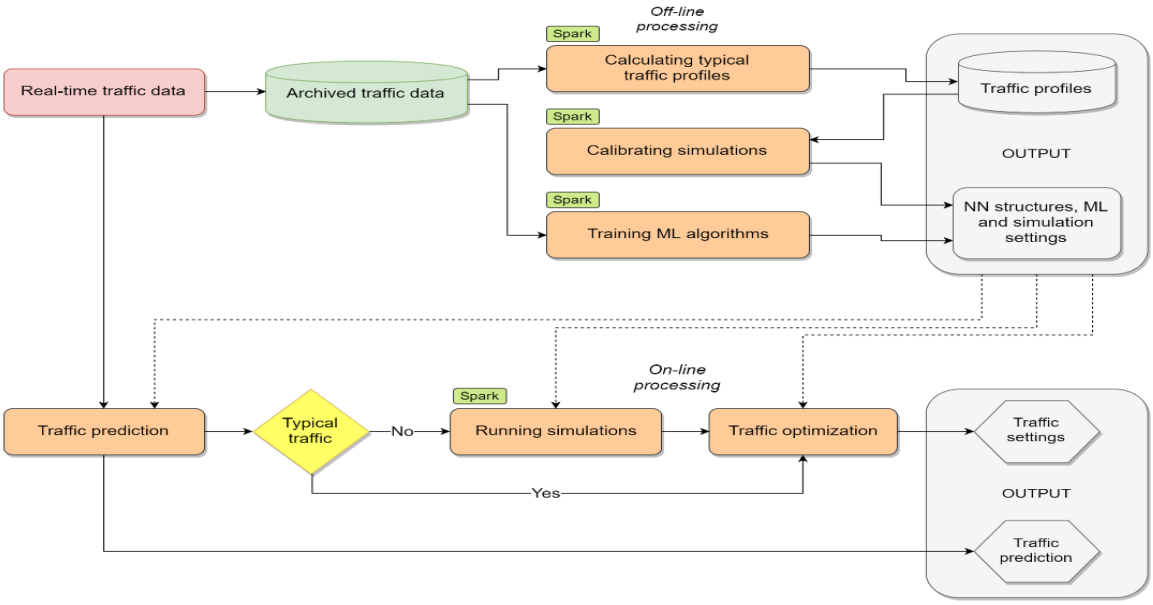


Figure 8 An architecture of the real-time traffic management system

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4.3.1.2 Overall Business Environment

Table 34 Case 2 - Business environment

Business role		**Partner involved**
Deployed application user	Who is the final user of the deployed application?	Traffic management centre, researchers interested in traffic information
Application provider	Who is providing the application to be deployed?	CET
Cloud provider	Who is providing the VMs?	Public or private cloud providers
Melodic platform user	Who is starting deployment execution and provide the model?	End user of the application or CET
Melodic platform administrator	Who is administrating the platform?	End user of the application or CET

4.3.1.3 Expected Benefits

Deployed application user: Traffic management centre

Table 35 Case 1 - Expected benefits for deployed application users

Benefit type	**Benefits description**
Speed	• Controlled computing time of selected tasks. • Manageable performance of the application. • Faster reaction and adaptation.
Cost	• Operations of traffic management system may be cheaper thanks to optimal configuration and only necessary resources used.
Reliability	• Stable performance of the system. • Increased availability.
Flexibility	• No-vendor lock-in, greater flexibility in choosing cloud provider.
Quality	• High overall quality of the system and implementation environment.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Application provider: CET

Table 36 Case 1 - Application provider expected benefits

Benefit type	Benefits description
Speed	• Shorter time of testing and deploying of the application.
Cost	• Decreased cost of testing and deploying of the application.
Reliability	• Offering highly available solution to the customers.
Flexibility	• No-vendor lock-in, greater flexibility in choosing cloud offering according to the customer requirements.
Quality	• High overall quality of the system and implementation environment.

Cloud provider: Public or private cloud provider

Table 37 Case 2 - Cloud provider expected benefits

Benefit type	Benefits description
Cost	• Decreased cost of the inhouse resources (private cloud providers).
Earnings	• Increased earnings. Thanks to easy and automatic deploy of the application to the public cloud Melodic may attract more customers.
Flexibility	• Avoiding overload of the private cloud.
Reliability	• Offering highly available solution to the customers.

Melodic platform user: End user of the application or CET

Table 38 Case 2 - Platform user expected benefits

Benefit type	Benefits description
Speed	• Shorter time of testing and deploying of the application.
Cost	• Decreased cost of testing and deploying of the application.
Reliability	• Offering highly available solution to the customers.
Flexibility	• No-vendor lock-in, greater flexibility in choosing cloud offering according to the customer requirements.
Quality	• High overall quality of the system and implementation environment.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Melodic platform administrator: End user of the application or CET

Table 39 Case 2 - Platform administrator expected benefits

Benefit type	Benefits description
Speed	• Finding problems faster thanks to platform monitoring.
Cost	• Reduced administration cost thanks to platform monitoring.
Reliability	• Reliability ensured by Melodic may simplify administration.
Flexibility	• Unified way of deploying various applications.
Quality	• Platform monitoring and other Melodic's tools may improve the quality of administration.

4.3.2 Melodic Individual User Roles

Table 40 Case 2 - Individual user roles

Melodic generic role	Use case specific role name	Description of the role in the use case (task, working environment, …)
System administrator (responsible for the initial installation of Melodic)	Administrator	Person responsible for: • initial installation of Melodic • monitoring and maintenance of Melodic • solving technical issues with Melodic
Application model provider (providing application data into CAMEL)	Developer/ CAMEL developer	Knows application very well and is responsible for describing application model and providing application data in CAMEL.
Deployment rules provider (utility function and overall deployment constraints)	Developer / CAMEL developer	Knows application as well as customer requirements including optimisation goals and is responsible for providing deployment rules in CAMEL.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Dataset provider (providing cloud offers)	Administrator / CAMEL developer	Person responsible for providing cloud offers rules in CAMEL.
Melodic end-user (running operational deployment)	Administrator	Person responsible for: • installation and launch of the application • monitoring the application and solving technical issues
Application end-user (using the deployed application)	Customer	Traffic engineer or operator in the traffic malmanagement centre or researcher interested traffic analysis.

4.3.3 Evaluation Groups

Following is a preliminary list of evaluators.

Table 41 Case 2 - Preliminary list of evaluation group members

Last name	First name	Profile	Company Unit	Specific role(s) in the scenario
Przeździęk	Tomasz	BM	CE-Traffic	Business and product manager, evaluating business impact
Novobilský	Jiří	BM	CE-Traffic	Business and product manager, evaluating business impact
Gora	Paweł	dev/adm	CE-Traffic	Designing and developing application
Vlčinský	Jan	dev/adm.	CE-Traffic	Real-time systems admin, developing application
Jarmuż	Dominika	data/dev/adm	CE-Traffic	Mobility data specialist, preparation of datasets, testing and evaluating application
Tatar	Kinga	data/dev	CE-Traffic	Mobility data specialist, preparation of datasets, testing and evaluating application
Stec	Katarzyna	data/dev	CE-Traffic	Junior mobility data specialist

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4.3.4 Applications to be deployed

The application's goal is to analyse traffic data to identify traffic patterns, make traffic predictions and, finally, optimise traffic.

Traffic data (e.g., FCD) are stored in a database and used for finding profiles of traffic (e.g., free flow traffic, typical traffic, traffic jam) for different areas and time periods. For each profile, the Traffic Simulation Framework (TSF) may be calibrated to simulate / predict traffic in given conditions. Archived data may be also used to train traffic prediction algorithms based on machine learning (e.g., recurrent neural networks and convolutional neural networks). Also, TSF may run simulations with different input settings and produce output, such as congestion, travel times, average speeds, total waiting times, so it may be used to evaluate many traffic control settings, e.g., traffic signal settings.

However, as the number of control settings is large, it is not easy to find the optimal one. To this end, the proposed approach takes advantage of AI / machine learning methods, for which TSF can be used to evaluate a large number (e.g., 100 000) of traffic control settings for each meaningful traffic conditions profile, in parallel, in a computational cluster. These evaluations are later used to train machine learning algorithms approximating outcomes of traffic simulations very fast (a few orders of magnitude faster than by running simulations) and with a very good accuracy (up to 99%). Thus, it may be possible to rapidly evaluate even larger sets of traffic control settings. For that purpose, the system may use the TensorTraffic tool. To find the best possible settings, the system may apply metaheuristics, e.g., genetic algorithms (again, Spark / Hadoop may be useful to parallelise computations).

The above approach may work well for finding optimal traffic signal settings for typical, recurrent traffic conditions, so it can be applied offline on meaningful traffic profiles (hence, the time of running computations is not crucial) producing default settings.

For the real-time traffic management, the system should analyse the current traffic state and predict the future state. If the detected or predicted state is too different from typical traffic profiles, real-time traffic data and TSF should be used to find new optimal traffic control settings.

The live traffic management system will consist of the following components:

- TrafficPredict – application for prediction and detection of typical traffic conditions,
- TrafficSimulate – application for traffic simulation based on TSF (also used for generation of training sets for TensorTraffic),
- TensorTraffic – application for approximating outcomes of traffic simulations,
- TrafficOptimise – application for finding optimal traffic lights settings.

The other components which will be used:

- database for storing archived traffic data,

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

- TrafficML – application for training machine learning algorithms,
- TrafficCalibrate – application for calibrating TSF.

4.3.4.1 Data Intensive Aspects

The first data intensive task is to infer typical traffic profiles and train traffic prediction models using archived floating car data. To do that it is necessary to process the large datasets originating from a period ranging from a several weeks to a several months and covering road segments within considered city or agglomeration. Since the data frequency is also high (1 minute) the required computations are data intensive.

Inferred typical traffic profiles will be used to calibrate TSF which is also a data intensive and compute intensive task. Later, calibrated TSF may generate training sets for machine learning algorithms for approximating outcomes of simulation. These algorithms require large training sets, and since they should run for many different areas and traffic profiles, the respective computations should be considered as data intensive.

4.3.4.2 Structural Application Model

Table 42 Case 2 - Structural application model

Criteria	Value
Machine learning applications	
RAM	>= 8 GB
CPU	>= 2 Cores
OS	Ubuntu
GPU	CUDA supported
Other applications	
RAM	>= 2 GB
CPU	>= 2 Cores, at least 2.5 GHz
OS	Ubuntu

4.4 Use case 3: Secure data management

7bulls is an SME provider of middleware for different kinds of organisations. The interests of 7bulls clients will be represented by FCR, a company offering SaaS solutions for secure document management with full access/flow control and approving operations with digital signature. The FCR team stems from the banking sector, mainly managers and security architects, making this solution compliant with the strictest security regulations. The system has been so far

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

implemented in finance and medical sectors, where there are large security requirements and the amount of communication is massive. The companies like FCR are interested in improving their services, in terms of efficiency and offering a level of security suitable for banks and other financial institutions. For now, this solution relies on a private cloud but it could actually benefit from the diversification of resources by using a public cloud with no vendor lock-in.

From the track record of cooperation with infrastructure providers, 7bulls believes that the cloud operators are not necessarily against the clients sharing their resources between different clouds. For example, they are willing to open the access to their resources to large clients that generally use private cloud, but peak moments they desire to use the public cloud in an easy and secure way. There is a variety of smaller infrastructure providers operating on the local markets, looking for the added value that would let them maintain their position in the market niche competing with the giants like Amazon. For demonstrating this model, the interests of this group will be represented by Dataspace, a Polish company delivering IT infrastructure as a service. Looking for advantages for letting them stay in the game, companies like Dataspace are willing to integrate with other operators to offer the value added services on top of the infrastructure to serve certain use cases that are not served by big players and offer them at a better rate.

4.4.1 Overview

This use-case has a specific focus on value added services for SMEs deployed on top of the cloud infrastructure. 7bulls will first gather the requirements and plan the deployment of value added services on the Melodic framework and then execute this scenario. It will be realized with the participation of an end user (FCR - a company cooperating with 7bulls, providing a SaaS solution for secure document management), besides a selected cloud services provider. The main goal is to prepare the Melodic framework for commercial use by implementing the ability to efficiently allocate the real-life cloud application to run using different cloud providers, to smoothly move the application from one provider to another and to be able to compare cost of running applications using different cloud providers. The consecutive iterations (due in month 18, 27 and 33) will deliver feedback to monitor the progress following the Evaluation framework defined in Task 5.1. The final output of the task will be a fully functional demonstration prototype of the FCR application integrated with Melodic to handle the operations described above.

4.4.1.1 Technical Architecture

The architecture as-is of FCR application is a typical three-tier architecture, with the following tiers:

- Client side: a Java applet running within a web browser.
- Server side: Java, Spring, Spring Boot, Spring security.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

- SQL database (Postgres, MS SQL, Oracle) and documents stored in the file system in an encrypted form.

Two types of data are being used in the application.

- Data stored in the relational database: users, permissions, index and metadata for documents, structure, auditing and so on.
- Data stored in the file system: encrypted documents.

The server-side part of the application could be scaled horizontally and there could be many instances of the server-side part of the application. This application element is compute intensive due to the many cryptographic operations. The current architecture of the FCR application is as follows:

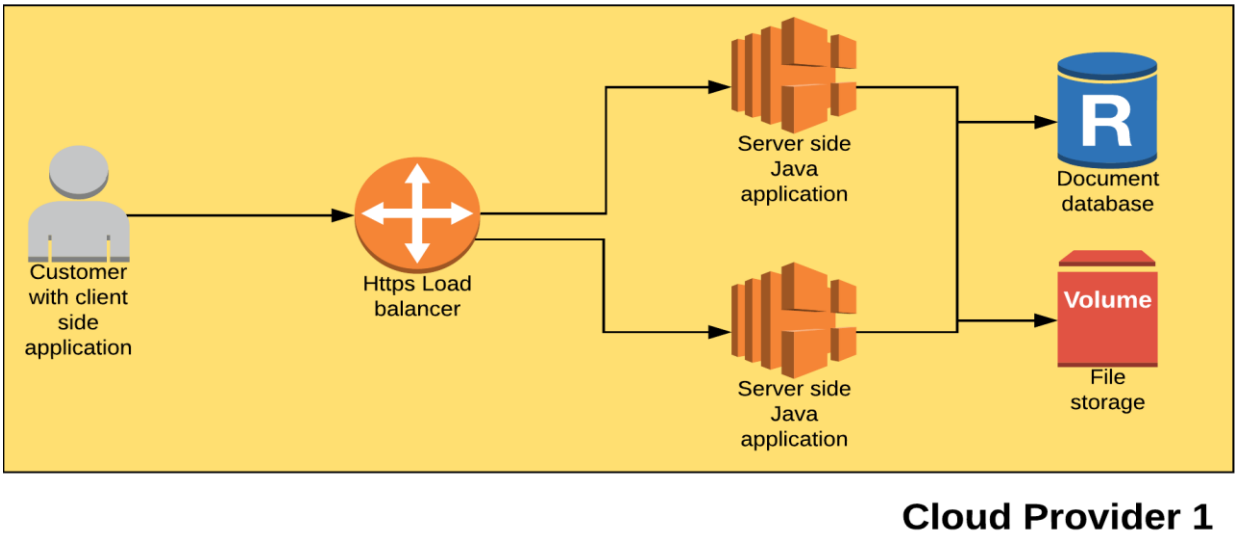


Figure 9 Case 3 - Technical architecture

Melodic will provide an easy to use Multi-Cloud environment for the benefits of both infrastructure providers and innovative SMEs deploying data-intensive applications in the Cloud. The document management solution offered by FCR requires Multi-Cloud to provide a highly secure processing of big data for mass communication of a financial institution with its customers. It could actually benefit from diversifying the resources by using both private and public Cloud. With Melodic, it could be a semi-automatic operation. FCR will benefit by optimising costs (by scaling private clouds to typical, not expected maximal loads) and eliminating vendor lock-in (using a number of public clouds and not just a single cloud). This model will work for any organisation using its own private Cloud and has a large commercial potential.

For this use case, we anticipate to use Melodic for scalability of the server-side components using the Scalability Rule Language (SRL) part of CAMEL. We plan to optimise the cost of the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

infrastructure using Melodic. In addition, the ability of Multi-Cloud application deployment will be an advantage for the FCR application. The figure below provides the anticipated architecture of the FCR application deployed on the Melodic framework.

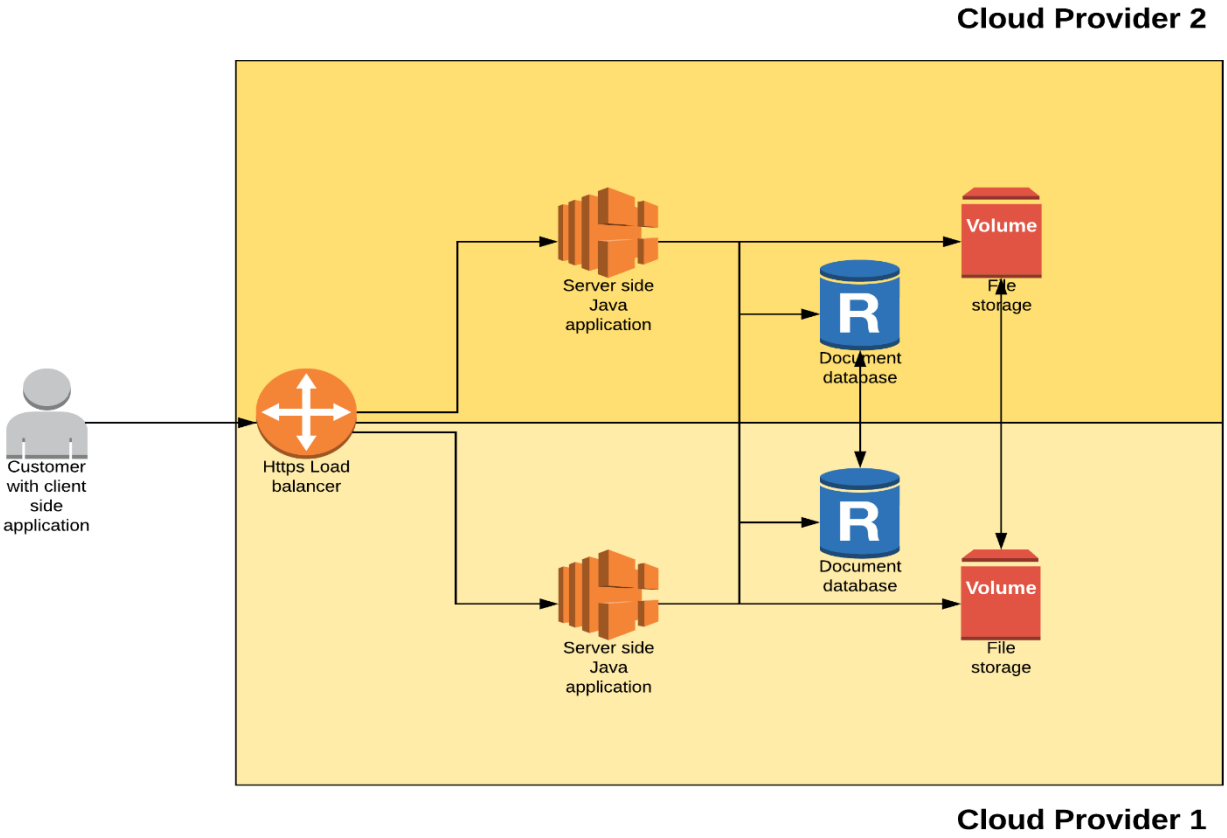


Figure 10 Case 3 - Technical architecture

4.4.1.2 Overall Business Environment

Table 43 Case 3 - Business environment

Business role		**Partner involved.**
Deployed application user	Who is the final user of the deployed application?	Any institution that processes large amounts of documents
Application provider	Who is providing the application to be deployed?	7bulls.com
Cloud provider	Who is providing the VMs?	Cloud providers: AWS, Azure, GCCP

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Melodic platform user	Who is starting deployment execution and provide the model?	End user and owner of the application or 7bulls.com
Melodic platform administrator	Who is administrating the platform?	End user and owner of the application or 7bulls.com

4.4.1.3 Expected Benefits

For now, this solution relies on a private cloud but it could actually benefit from the diversification of resources by using both private and public clouds with no vendor lock-in. Melodic will provide an easy to use multi-cloud environment for the benefits of both infrastructure providers and innovative SMEs deploying data-intensive applications in the Cloud. With Melodic, it could be a semi-automatic operation. FCR will benefit by optimising costs, scaling private clouds to typical, not expected maximal loads, and eliminating vendor lock-in (using a number of public clouds). Melodic will offer customers like FCR the increased value by allowing them to:

1. Compare different cloud services provider offers
2. Smoothly move from one cloud service provider to another in case of a better value for a given price
3. Choose any cloud service provider compatible with Melodic (no vendor-lock)
4. Create disaster recovery installation for applications using different cloud services providers

Deployed application users

Table 44 Case 3 - Expected benefits for deployed application users

Benefit type	Benefits description
Speed	• Shorter time of app deployment and computing time
Cost	• More efficient cost calculation
Reliability	• More reliable method of choosing providers and configuration of app specific deployment
Flexibility	• More flexible way of choosing provider for app deployment
Quality	• Implementation environment of the application with the highest quality available at the moment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Application provider: 7bulls.com

Table 45 Case 3 - Application provider expected benefits

Benefit type	Benefits description
Speed	• Shorter time of deployment and installation
Cost	• Decreased cost of deployment and installation
Reliability	• More reliable method of choosing providers and configuration of app specific deployment
Flexibility	• More flexible way of choosing provider for app deployment
Quality	• Increased quality of app operation

Cloud provider: AWS, Azure, GCCP

Table 46 Case 3 - Cloud provider expected benefits

Benefit type	Benefits description
Speed	• Delivering only resources that fit best in each case
Cost	• Optimization of the infrastructure consumed • More resources to be sold
Reliability	• Increase customer confidence by providing solutions tailored to their needs
Flexibility	• Better management of own resources
Quality	• Providing only those resources that are adequate to the customer's expectations

Melodic platform user: End user and owner of the application or 7bulls.com

Table 47 Case 3 - Platform user expected benefits

Benefit type	Benefits description
Speed	• Increased speed of app deployment
Cost	• Decreased cost of app deployment and usage

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Reliability	• More reliable and suitable method of choosing providers and configuration of app specific deployment
Flexibility	• Better management of resources used for app
Quality	• Implementation environment of the application with the highest quality available at the moment

Melodic platform administrator: End user and owner of the application or 7bulls.com

Table 48 Case 3 - Platform administrator expected benefits

Benefit type	Benefits description
Speed	• Increasing speed of deploying apps because of standardized way of such deployments
Cost	• Reducing costs of used resources
Reliability	• Reliance on two independent sources determining the range of resources needed (CAMEL and service provider)
Flexibility	• Unified way of deploying apps
Quality	• Using common standards of quality checking in specific cases

4.4.2 Melodic Individual User Roles

Table 49 Case 3 - Individual user roles

Melodic generic role	Use case specific role name (i.e. the name that the partner will use)	Description of the role in the use case (task, working environment, ...)
System administrator (responsible for the initial installation of Melodic)	Admin	• Initial installation • Launching application • Monitoring • Maintenance
Application model provider (providing application data into CAMEL)	Developer/Camel developer	• Providing input data • Uploading camel file

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Deployment rules provider (utility function and overall deployment constraints)	Developer/Camel developer	• Providing input data • Uploading camel file
Dataset provider (providing cloud offers)	Developer/Camel developer	• Providing input data • Uploading camel file
Melodic end-user (running operational deployment)	Admin	• Initial installation • Launching application • Monitoring • Maintenance
Application end-user (using the deployed application)	Customer	• Case specific

4.4.3 Evaluation Groups

Following is a preliminary list of evaluators.

Table 50 Case 3 - Preliminary list of evaluation group members

Last name	**First name**	**Profile (dev., adm. or BM)**	**Company Unit**	**Specific role(s) in the scenario**
Skrzypek	Paweł	BM/adm	7bulls.com	Architect, Business Manager and business impact evaluation
Kowalski	Grzegorz	adm	7bulls.com	DevOps
Prusinski	Marcin	dev	7bulls.com	Developing the app
Szkup	Paweł	dev	7bulls.com	Developing the app
Różanska	Marta	dev	7bulls.com	Developing the app
Bankowska	Edyta	test	7bulls.com	Testing and evaluating the app
Materka	Katarzyna	BM	7bulls.com	Business impact evaluation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Semczuk	Michał	BM	7bulls.com	Business impact evaluation

4.4.4 Applications to be deployed

The FCR Secure Document Management application is a SaaS solution for secure document management with full access/flow control and approving operations with digital signature. The FCR application is designed mainly for the banking sector and is compliant with the strictest security regulations. The system has been so far implemented in finance and medical sectors, where there are large security requirements and the amount of communication is massive. Companies like FCR are interested in improving their services, in terms of efficiency and security (the level of security suitable for banks and other financial institutions). For now, this solution relies on a private Cloud but it could actually benefit from the diversification of resources by using public Cloud with no vendor lock-in.

4.4.4.1 Data Intensive Aspects

FCR is a provider of SaaS solution for secure document management with full access/flow control and approving operations with digital signature. The system has been so far implemented in finance and medical sectors, where there are large security requirements and the amount of communication are massive. This solution struggles with enormous number of files, which are collected and stored by institutions. The companies like FCR are interested in improving their services, in terms of efficiency and security (the level of security suitable for banks and other financial institutions).

4.4.4.2 Structural Application Model

Table 51 Case 3 - Structural application model

Criteria	Value
RAM	> 4 GB
CPU	> 2 core
OS	Ubuntu x64

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4.5 Use case 4: Genome analysis

7bulls will adapt the Melodic framework to process sensitive data that are easy to be partitioned and anonymised by splitting both data and metadata. Due to recent advances in genome and protein research, 7bulls has been facing a growing interest in genome/protein data processing from innovative start-ups, SMEs and small research groups inside larger organizations, both public and private. Such data processing involves large data that is usually extremely sensitive with the most sensitive medical data on specific patients. At the same time, this data is easy to partition and anonymise, not to mention the cryptographic measures. Managing the cost and time of such analysis would be beneficial. Some of them, like choosing optimal method for specific patients, must be done in a short time and with reasonable costs known in advance, while others like research on new general methods of treatments can be done in a longer period with aggressive cost optimisation.

In this case, Melodic will enable 7bulls to exploit a completely new market segment. Sensitive data processing presents a fast-growing market, with an increasing number of organisations interested in this kind of services, from big pharma companies focusing on flexibility, scalability and security to start-ups or academic research groups that appreciate low entrance and operational costs as well as no vendor lock-in. By cooperating with academia (University of Białystok and several research groups in medical universities) as well as selected business partners, 7bulls has access to this market. 7bulls cooperates with a team of people with academic background and has contacts with researchers that could work on the scientific aspects of sensitive data processing, offering access to concrete tools and methods of genome data analysis.

7bulls will use the Melodic framework to develop plugins handling the data/metadata separation, data partitioning/distribution with proper security mechanisms, as well as provide support for processing such data on distributed nodes and gathering results. Under the Melodic project, this technology will be validated for implementation supporting the processing of genome.

4.5.1 Overview

This task is focused on using Melodic for processing partitionable sensitive data. 7bulls will cooperate with the bioinformatics research group at the University of Bialystok (PL) to verify and demonstrate how Melodic will enable the distributed processing of the genome data, which come in a large volume and is usually extremely sensitive. At the same time, this data is relatively easy to partition, anonymize, and protect through cryptographic measures. The demonstration will show how Melodic can be used to manage the cost and time of genome data analysis. 7bulls will develop and verify in iterative cycles a set of specific plugins to support such operations and further processing of genome, protein data sets. The final output of the task will be a fully

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

functional demonstration prototype of the genome analysis application integrated with Melodic to handle the operations described above.

4.5.1.1 Technical Architecture

Selection of the appropriate data for analysis is one of the important success factors of this use case. Initially, we will focus on open, publicly available datasets of genome sequence or genome expression data. One of the considered possibilities is the data from 1000 Genome project. Data will be stored in the flat file. The architecture of the application will be based on the following frameworks:

- Spark - fast and generic engine for distributed, large-scale data processing;
- Mesos (optionally) - cluster resource management system that provides efficient resource isolation and sharing across distributed applications;
- Nvidia CUDA - technology for GPU parallel computing.

A simplified application architecture is presented in Figure 11. It should be noted that this is the initial concept that might evolve in the later stages of development.

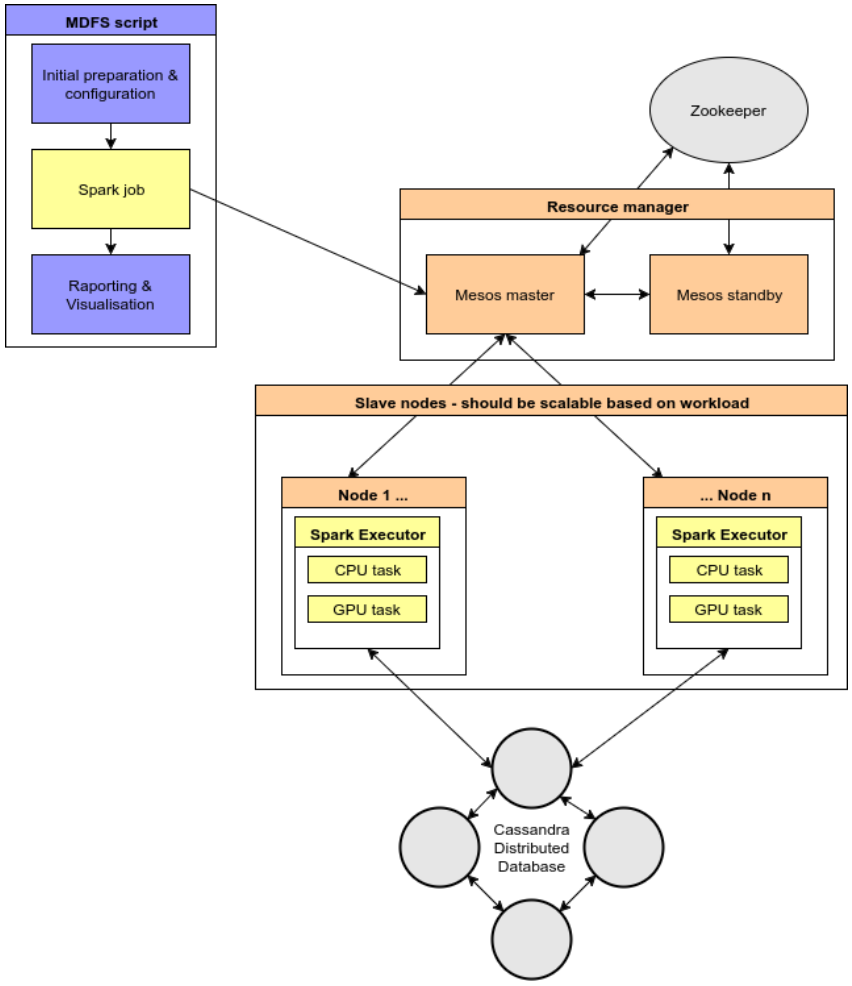


Figure 11 Case 4 - Technical architecture based on frameworks

4.5.1.2 Overall Business Environment

Table 52 Case 4 - Business environment

Business role		Partner involved.
Deployed application user	Who is the final user of the deployed application?	Universities, hospitals and other and other institutes processing genomic data
Application provider	Who is providing the application to be deployed?	7bulls.com
Cloud provider	Who is providing the VMs?	Cloud providers: AWS, Azure, GCCP
Melodic platform user	Who is starting deployment execution and providing the model?	End user and owner of the application or 7bulls.com
Melodic platform administrator	Who is administrating the platform?	End user and owner of the application or 7bulls.com

4.5.1.3 Expected Benefits

7bulls will adapt Melodic to process sensitive data that is easy to be partitioned and anonymised by splitting data and metadata: genome data sets. Managing the cost and time of analysis would be very useful - some of them (like choosing optimal method for specific patients) must be done in a short time and with reasonable costs known in advance. Others (like research on new general methods of treatments) can be done in a longer period with aggressive cost optimization. Depending on SME interests, 7bulls will develop plugins in Melodic to handle data/metadata separation, data partitioning/distribution with proper security mechanisms, support for processing it on distributed nodes and gathering results.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Deployed application users

Table 53 Case 4 - Expected benefits for deployed application users

Benefit type	Benefits description
Speed	• Shorter time of app deployment and execution time
Cost	• More efficient cost calculation
Reliability	• More reliable method of choosing providers and configuration of app specific deployment
Flexibility	• More flexible way of choosing provider for app deployment
Quality	• Implementation environment of the application with the highest quality available at the moment

Application provider: 7bulls.com

Table 54 Case 1 - Application provider expected benefits

Benefit type	Benefits description
Speed	• Shortened time of installation
Cost	• Decreased cost of installation
Reliability	• More reliable method of choosing providers and configuration of app specific deployment
Flexibility	• More flexible way of choosing provider for app deployment
Quality	• Increased quality of app operation

Cloud provider: AWS, Azure, GCCP

Table 55 Case 4 - Cloud provider expected benefits

Benefit type	Benefits description
Speed	• Delivering only resources that fit best in each case
Cost	• Optimization of the infrastructure consumed • More resources to be sold

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Reliability	• Increasing customer confidence by providing solutions tailored to their needs
Flexibility	• Better management of own resources
Quality	• Providing only those resources that are adequate to customer's expectations

Melodic platform user: End user and owner of the application or 7bulls.com

Table 56 Case 4 - Platform user expected benefits

Benefit type	Benefits description
Speed	• Increased speed of app deployment
Cost	• Decreased cost of app deployment and usage
Reliability	• More reliable method of choosing providers and configuration of app specific deployment
Flexibility	• Better management of resources used for app
Quality	• Implementation environment of the application with the highest quality available at the moment

Melodic platform administrator: End user and owner of the application or 7bulls.com

Table 57 Case 4 - Platform administrator expected benefits

Benefit type	Benefits description
Speed	• Increasing speed of deploying apps because of standardized way of such deployments
Cost	• Reducing costs of used resources
Reliability	• Reliance on two independent sources determining the range of resources needed (CAMEL and service provider)
Flexibility	• Unified way of deploying apps
Quality	• Using common standards of quality checking in specific cases

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4.5.2 Melodic Individual User Roles

Table 58 Case 4 - Melodic Individual User Roles

Melodic generic role	**Use case specific role name (i.e. the name that the partner will use)**	**Description of the role in the use case (task, working environment, ...)**
System administrator (responsible for the initial installation of Melodic)	Admin	• Initial installation • Launching application • Monitoring • Maintenance
Application model provider (providing application data into CAMEL)	Developer/Camel developer	• Providing input data • Uploading camel file
Deployment rules provider (utility function and overall deployment constraints)	Developer/Camel developer	• Providing input data • Uploading camel file
Dataset provider (providing cloud offers)	Developer/Camel developer	• Providing input data • Uploading camel file
Melodic end-user (running operational deployment)	Admin	• Initial installation • Launching application • Monitoring • Maintenance
Application end-user (using the deployed application)	Customer	• Each case specific

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4.5.3 Evaluation Groups

Following is a preliminary list of evaluators.

Table 59 Case 4 - Preliminary list of evaluation group members

Last name	First name	Profile (dev., adm. or BM)	Company Unit	Specific role(s) in the scenario
Skrzypek	Paweł	BM/adm	7bulls.com	Architect, Business Manager and business impact evaluation
Kowalski	Grzegorz	adm	7bulls.com	DevOps
Prusinski	Marcin	dev	7bulls.com	Developing the app
Szkup	Paweł	dev	7bulls.com	Developing the app
Różanska	Marta	dev	7bulls.com	Developing the app
Bankowska	Edyta	adm	7bulls.com	Testing and evaluating the app
Materka	Katarzyna	BM	7bulls.com	Business impact evaluation
Semczuk	Michał	BM	7bulls.com	Business impact evaluation

4.5.4 Applications to be deployed

With the advent of the "Omics" era in the life sciences, researchers gained access to vast amounts of biological information, including data about genome, proteome, metabolome, transcriptome and molecular pathways just to name a few. The size of this data has now exceeded well beyond petabytes or even Exabytes. As an example, the final results from 1000 Genome Project have a size of more than 200 terabytes of data. The forthcoming initiatives, like the 100,000 Genome Project27, give strong indications that the amount of data available for analysis will grow exponentially. To take full advantage of this data, scientists and developers will need to develop tools and platforms that will enable them to perform calculations on a scale well beyond a small cluster.

As a part of Melodic use-case, our research team will develop an application prototype that enables a robust approach for the discovery of synergistic variables in biological datasets, with a main focus on data from gene expression studies and genome-wide association study (GWAS).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

These datasets are often described with a large number of variables. Only few of those variables are usually relevant for the phenomena under the researcher's investigation. Therefore, the identification of variables that are relevant for a given research is an important initial step of data analysis. The common way to identify the relevant variables is a univariate test for association between each explanatory variable and the response variable, but the univariate test ignores variables, which contribute information on the response only in synergy with others.

To successfully detect such relevant variables, it is necessary to examine all the k-tuples of variables. The multivariate exhaustive search requires huge amount of computations, which have been impossible since a long time.

Table below shows the number of tests required for 50 000 variables

k	**Number of Tests**
1	$5.0 * 104$
2	$1.2 * 109$
3	$2.0 * 1013$

To overcome this stalemate situation, we will develop a prototype application that:

- can benefit from Cloud computing processing power and scalability (by complying to the Melodic application model)
- can make use of Graphics Processing Units (CUDA technology) for speeding up calculations
- applies a novel algorithm developed by the Faculty of Computer Science of Bialystok University, tailored for this problem

4.5.4.1 Data Intensive Aspects

Complex analysis of big genome data is usually extremely sensitive (most sensitive medical data on specific patients); at the same time, this data is easy to partition and anonymise, not to mention their encryption through applying respective cryptographic measures. Managing the cost and time of analysis would be very useful - some of them (like choosing optimal method for specific patients) must be done in a short time and with limited costs known in advance, others (like research on new methods of treatments) can be done in a longer period with cost optimization.

It is a fast-growing market, with growing number of organisations interested in this kind of services, from big pharma focusing on flexibility, scalability and security to start-ups or academic research groups that appreciate low entrance and operational costs and no vendor lock-in.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

By cooperating with academia and selected business partners, we have access to this market. We have a team of people with academic background and contacts with researchers that could work on the scientific aspects of this (offering access to concrete tools / methods of genome data analysis).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

5 Next Steps

5.1 Questions and Metrics

In the scope of D6.2 and D6.3, the questions and the related metrics will be defined. The provided plan will be a roadmap that contains indicative goals, questions, and metrics for executing the validation as defined in the previous steps. The plan will serve as a guideline for all involved actors and will provide the basis for the subsequent measurement plans and the analysis actions.

5.2 The Data Collection Phase

The data collection procedures of the Melodic Evaluation Framework will include the way in which procedures are defined, the way in which data collection forms are applied, and the way in which tools support the data collection process.

Usually, during the data collection phase, manual, electronic and automated data collection procedures can be employed. In the Melodic evaluation process, electronic data collection forms will be used to automatically handle the data entry activities and will comprise an efficient way of collecting data. Despite the fact that electronic forms require similar effort when compared to the manual forms, their advantage is that the data does not have to be re-typed into a measurement database. Electronic forms are a clear improvement over manual forms, as manual forms should be continuously available, distributed and updated.

According to the timetable presented in Table 15, this phase will take place by the end of the project in the scope of the preparation of D6.4.

5.3 The Interpretation Phase

This final phase of the Evaluation Framework will include all activities required to actually curate, store and process the measurement data. In the data interpretation phase, the focus will be shifted to drawing conclusions regarding the results of the measurement scheme. The conclusions are usually specific for each object under validation. This is an essential phase since this step tries to find answers to the questions underlying the measurement scheme.

The interpretation phase mainly concerns processing the collected data into presentable and interpretable material. The GQM plan provides the basis for preparing feedback sessions: feedback material should support answering the questions as defined in the Evaluation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Framework, and based on these answers, one should be able to conclude whether the defined measurement goals are attained. This process has to be done for each goal under validation.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

6 Conclusion

This is the first deliverable of WP6. WP6 as a whole is responsible for the use cases and the evaluation of the Melodic platform. This deliverable documents the Melodic evaluation framework, the validation scenario definitions as well as the organization and planning of the use case demonstrations.

Based on the framework and related methodologies presented in this deliverable, the universal applicability of Melodic will be demonstrated by four use cases delivered by Melodic partners relying on big data technologies, but working with different technical constraints and business models.

The implementation feasibility of these utility features and the related liability of the utility-based deployment will be evaluated. In order to ensure that they are ready to be deployed by the integration release, the applications to be used in the use cases will be implemented and modelled in the new CAMEL release extended with the big data aspect in the scope of the next deliverable (D6.2). The implementation feedback presented there will flow back to WP2-WP5 to improve and extend the feature set as new Melodic releases are made available. The use cases themselves will be a crucial argument by the end of the project to demonstrate Melodic to real potential customers thus preparing commercial use of the Melodic platform beyond the end of the project.

The final evaluation will provide answers to the following core expectations for Melodic

1. Does the Melodic platform operate and do what it is required to perform?
2. Does a Melodic application run correctly (functional and non-functional)?
3. Does a Melodic application run more efficiently (cost, elapsed time and green)?
4. Is the cost of using Melodic justified by its benefits?

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

7 References

[1] V. R. Basili, , G. Caldiera i H. D. Rombach, „THE GOAL QUESTION METRIC APPROACH," w *Encyclopedia of Software Engineering, 2 Volume Set*, 1994.

[2] P. Skrzypek, S. Kicin, K. Materka, A. Schwichtenberg, S. Mazumdar, J. Domaschka, Y. Verginadis, M. Semczuk i S. Schork, „Integration and Testing Requirements (D5.04)," 2017.

[3] S. H. Kan, Metrics and Models in Software Quality Engineering, Addison-Wesley Professional, 2002.

[4] „ISO 25000 Portal," [Online]. Available: http://iso25000.com/index.php/en/iso-25000-standards/iso-25010. [Data uzyskania dostępu: 09 February 2018].

[5] J. Nielsen, Usability Engineering, Elsevier, 1994.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Annex 1 - Evaluation Components (based on WP5 test scenarios)

Initial deployment

- Installation and deployment of a N-component application on M different Cloud Providers
- Installation and deployment of a N-component application in Docker containers on M different Cloud Providers
- Installation and deployment of a N-component application, where X component are installed in a Docker container and Y on a normal VM on M different Cloud Providers
- Deployment requirement enforcement.
- Installation and deployment of a N-component application on M different Cloud Providers with more advanced set of requirements, like non-functional ones.

Metric management

- Built-in raw metrics collection
- Custom raw metrics collection
- Composite metric collection
- Event generation

Local reconfiguration

- Scale out application
- Scale in application

Global reconfiguration

- Attributes of used VM offerings changed
- Global reconfiguration

Reasoning

- Linear constraints and optimization solving - CP Solver
- Linear constraints and optimization solving - MILP Solver
- Linear constraints and optimization solving - LA Solver
- Non-linear constraints and optimization solving - CP Solver
- Non-linear constraints and optimization solving - LA Solver

API

- Camel model upload

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

- Initiate deployment process
- Get application status

UI

Web based UI for application view:

- Application view
- Deployment view

Eclipse based editor of the CAMEL:

- CAMEL Model validation
- Syntax completion

BigData management

- Big data application deployment optimization
- Big data application deployment execution
- Big data application monitoring and reconfiguration
- Data locality awareness - features related to data locality and data movement.

Fault handling

- Temporary unavailability of Melodic platform components
- Temporary unavailability of BPM - verifying proper system behaviour after BPM recovery.
- Temporary unavailability of Cloud Provider
- High Availability Component configuration

Performance

- Response time while solving complex allocation problems
- Dynamic scalability within one Cloud - verification of the execution time
- Dynamic scalability testing for multi-Cloud feature (using two different locations)
- Counting Compute Resource Overhead of Melodic introduced over its host machine

Security

- Method invocation by programmatic access - Successful Authentication
- Unsuccessful authentication
- Successful Authorisation Request
- Unsuccessful authorisation request
- Unsuccessful user authorisation with administrator privileges
- Logging within Melodic platform

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

User management

- Adding user
- Removing user
- Updating user password
- Updating user profile
- Unified starting, stopping and restarting of Melodic platform
- Configuring backup
- Executing backup
- Recover Melodic platform
- Monitor Melodic platform

Annex 2 - Utility examples

Example 1: Combinatorics

Consider the problem of deploying an integral number V virtual machines of an integral number M machine types, and consider first the case where V is larger or equal to M. The way of writing an integer as a sum of term where the position of the term counts, i.e. 6 = 4 + 2 is considered different from 6 = 2 + 4 is called a *composition* in number theory, and the case where zero terms are allowed is called a *weak composition* and the count is given by

```
CompositionCount[ VirtualMachines_Integer?Positive, MachineTypes_Integer?Positive ]/;TrueQ[VirtualMachines>=MachineTypes]:= Binomial[ VirtualMachines + MachineTypes -1, MachineTypes - 1];
```

The actual configurations can be generated as solutions to the Frobenius equation

```
AllCompositions[ VirtualMachines_Integer?Positive, MachineTypes_Integer?Positive ]:= FrobeniusSolve[ ConstantArray[1,MachineTypes],VirtualMachines ]
```

If there are more machine types than virtual machines to be started, then it is possible to use only subsets of the machines. The number of subsets with k machines taken from M possible machines is the binomial coefficient

$$\binom{M}{k}$$

For each of these subsets, the number of virtual machines must be decomposed over the subset.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

It is therefore necessary to compose V of exactly k non-zero terms, since the machines not being use have already been excluded when selecting the subset. The number of compositions is given by the binomial coefficient

$$\binom{V-1}{k-1}$$

Consequently, the number of compositions is obtained by multiplying these two factors and adding over k representing the number of machine types to use. At most one may use as many machine types as there are virtual machines to be deployed.

```
CompositionCount[ VirtualMachines_Integer?Positive, MachineTypes_Integer?Positive ]/;TrueQ[ VirtualMachines < MachineTypes ] :=
  Sum[ Binomial[ MachineTypes, k] * Binomial[ VirtualMachines - 1, k - 1 ], {k, 1, VirtualMachines} ];
```

Examples

If one needs 6 virtual machines and these can be chosen from 3 different machine types, then the number of possible configurations is

```
CompositionCount[ 6, 3 ]
28
```

This is all of these configurations enumerated

```
AllCompositions[ 6, 3 ]

{{0,0,6},{0,1,5},{0,2,4},{0,3,3},{0,4,2},{0,5,1},{0,6,0},{1,0,5},{1,1,4},{1,2,3},{1,3,2},{1,4,1},{1,5,0},{2,0,4},{2,1,3},{2,2,2},{2,3,1},{2,4,0},{3,0,3},{3,1,2},{3,2,1},{3,3,0},{4,0,2},{4,1,1},{4,2,0},{5,0,1},{5,1,0},{6,0,0}}
```

and it can be confirmed that the count is correct by counting the number of elements in this set

```
Length[ AllCompositions[ 6, 3 ] ]
28
```

Alternatively, let there be 7 machine types to choose with the same 6 virtual machines to be allocated. The number of compositions is then

```
CompositionCount[ 6, 7 ]
924
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

It makes no sense to display them, but the number is not too large to be directly enumerated, and then counted to verify the above formula.

```
Length[ AllCompositions[ 6, 7 ] ]
924
```

Example 2: Secure Documents: Load distribution over workers

Problem description

The application consists of a file system storing a set of files, a database server indexing these files, and one or more application servers receiving connections from the users of the application. The application servers do complicated encryption and decryption on behalf of the users, and the load on these servers will depend on the number of users and, primarily, the size and content of the files they are accessing.

In order to ensure experienced application quality, it is desirable to scale horizontally by starting more application servers if the users' average response time T_R is large. There is an absolute requirement that $T_R < T_{max}=30$ s. For each document, an application level sensor will record the response time experienced to the users. This will be aggregated to the average response time for all users on that application server, which will again be aggregated to the average response time for all application servers in the system. Hence, $\overline{T}_R(k)$ is the measured average response time at sampling iteration *k*.

The expressed utility is formulated as: ***Deploy with minimal cost while keeping the average response time limited.***

Pricing model

It is difficult to define an exact *cost model*, however, given a machine type offered by a provider it will be possible to monitor the price for this machine at regular intervals, and the *cost* used in this model will simply be the number of instances of this machine type multiplied with the unit price for this machine type.

It should be noted that this is a linear pricing model, and consequently it will be additive: *cost(a+b) = cost(a) + cost(b)*. The cost of a number of instances of given machine type is therefore given as:

```
Cost[ MachineType_Symbol, Cardinality_Integer?NonNegative ]:= Cardinality * MachineType[Price];
```

The cheapest machine type can be selected based on the price alone. Since the function used can

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

take several of the smallest elements, it returns a list even if that list is supposed to have only one single element. It is therefore necessary to pick out the single element that is returned since the selection function is explicitly confined to the single smallest element.

```
SelectCheapest[ CandidateMachines_List?VectorQ  ] := TakeSmallestBy[ CandidateMachines, (#[Price])&, 1][[1]];
```

A ***deployment*** is a set of machines together with the cardinality of each type. The format of the configuration is a list of lists, where each list has two elements: The machine type followed by the cardinality. Given that the provided argument matches this format, computing the cost of the configuration is straightforward as it is simply applying the above cost function to each element and then add the results together.

```
Cost[ Deployment_List ] /; VectorQ[ Deployment, MatchQ[#,{_,_Integer?NonNegative}]&]:= Total[ Map[ Apply[ Cost, #]&, Deployment ]];
```

Example

For the toy example of this notebook only two machine types will be defined

```
NodeCandidates = {mBig, mXXL};
```

and their prices are

```
mBig[Price]^= 6;
mXXL[Price]^= 10;
```

The cost of using, say, 5 big machines and 2 extra-large machines is then given by

```
Cost[{{mXXL,2},{mBig,5}}]
55
```

while the cost of swapping the number of machines of the two types should be higher

```
Cost[{{mXXL,5},{mBig,2}}]
64
```

In total, this corresponds to 7 = 5 + 2 machines. Interestingly there are many ways 7 machines can be divided among the two machine types

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

```
machines = AllCompositions[7,2]
{{0,7},{1,6},{2,5},{3,4},{4,3},{5,2},{6,1},{7,0}}
```

If the first number corresponds to the big machine and the second to the extra-large machine, then the cost of these configurations can be computed as

```
Map[ Cost[{{mBig,#[[1]]},{mXXL,#[[2]]}} ]&,machines]
{70,67,64,61,58,55,52,49}
```

The configurations can be sorted according to to cost, which confirms the intuition that configurations with more big machines are less costly than configurations with more extra-large machines.

```
CostSortedDeployments = SortBy[ machines, Cost[{{mBig,#[[1]]},{mXXL,#[[2]]}} ]&]
{{7,0},{6,1},{5,2},{4,3},{3,4},{2,5},{1,6},{0,7}}
```

Utility dimension cost: Marginal cost approach

In general, adding another application server increases the cost, and *decreases* utility. However, one has to realise that the marginal cost of a new machine is dependent on the number of machines already started. Consider for instance that the current cost of the running machines is 1000€/hour. Adding a new machine adding, say, 10€/hour to this cost might not cause a huge change in utility.

Let D*(k)* be the current *deployment* running in iteration *k*, and let D(k+1) be the new deployment proposed at this step.The change in cost utility can be modelled by saying that the current cost utility is decremented by a factor for the change in marginal cost.

$$cost(D(k))/cost(D(k+1))$$

If the cost of the new configuration is higher than the current, this factor will be less than unity and give a reduced utility. However, if the cost of the next deployment is less than the current deployment the factor will be larger than unity and resulting in a higher utility, but it is necessary to ensure that the utility stays bounded in the interval [0,1]. This will be discussed in the next section.

Utility

The resulting utility, using the scale factor, may be defined as

$$U_{cost}(D(k+1)|D(k))=U_{cost}(D(k)\ |D(k-1))*cost(D(k))/cost(D(k+1))$$

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

The issue here is that the utility is not unique but conditioned on the full sequence of different deployments tried. Conceptually, a deployment is a node in a fully connected graph. The marginal cost factor can be seen as weights on the edges of this graph, and the utility assigned to a particular configuration depends on which vertex in the graph the configuration moved away from. The result is that the utility of a configuration is Markov process.
Had it been possible to enumerate all deployments, one could have calculated the utility as the expected utility for each deployment. In practice, one will need to define the utilities as the search algorithm of the solver tries different configurations, and not all configurations may be tried. One can therefore either give the utility of a configuration as the average of the utilities obtained by moving to the deployment based on this configuration, or use a solver that accepts a stochastic utility value, i.e. two successive proposals of the same configuration give two different utility values. This latter approach is assumed in the following implementation.
The benefit of this approach is that it is insensitive to the number of configurations. Had the cost been bounded, one could have use an absolute cost utility. However, even if the machine types possible will probably stay fairly constant, the total number of machines can be increased as needed, with the result that it is not possible to compute an upper bound on the cost (unless there is a constraint putting a cap on the highest possible cost).
With the above reasoning, one must first define the utility of the initial configuration to some reasonable element. A natural thing to start a deployment of this application would be to start with one virtual machine and the cheapest machine.

```
InitialDeployment = {{SelectCheapest[ AllowedMachines ],1}}
{{SelectCheapest[AllowedMachines],1}}
```

And the cost utility of this deployment is then set to perfect.

```
Utility[ MarginalCost, InitialDeployment ] = 1;
```

The cost utility function takes the new deployment based on the given configuration and the old deployment, and calculates and define the new utility value for the new configuration making sure that the utility is capped at unity.

```
Utility[ MarginalCost, CurrentDeployment_List, NewDeployment_List ]/;( VectorQ[ CurrentDeployment, MatchQ[#,{_,_Integer?Positive}]&] && VectorQ[ NewDeployment, MatchQ[#,{_,_Integer?Positive}]&] ) := ( Utility[ cost, NewDeployment ] = Min[ Utility[ MarginalCost, CurrentDeployment ]* Cost[ CurrentDeployment ] / Cost [ NewDeployment ], 1] );
```

Assume that the next deployment is one consisting of only one extra-large machine, then the cost utility will be

```
Utility[ MarginalCost, InitialDeployment, {{mXXL,1}} ]//N
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

```
0.3
```

The utility of a mixed deployment following this will be even worse, not surprising since it contains the same extra-large machine,

```
Utility[MarginalCost, {{mXXL,1}},{{mBig,1},{mXXL,1}}]//N
0.230769
```

and so one could try to make a deployment with two big machines

```
Utility[MarginalCost, {{mBig,1},{mXXL,1}},{{mBig,2}}]//N
0.5
```

Utility dimension cost: Cost bound

As an alternative to the preceding approach trying to view a new deployment based on a new configuration relative to the current deployment, one may try to estimate the cost utility directly. One approach is to say that the requested configuration is implemented only by using the cheapest possible machine type. This cost is then the *lower bound* for the average cost of a machine in the given configuration. The ratio between the lowest cost and the average cost could then be the utility. This will obviously be unity if only the cheapest machines are used in the new deployment candidate, and decrease with the increasing cost of the machines without forcing an upper bound on the allowed cost.

```
Utility[ CostBound, NewDeployment_List ]/;( VectorQ[ NewDeployment, MatchQ[#,{_,_Integer?Positive}]&] ) := With[
  { minprice = Min[Map[(#[Price])&,NodeCandidates]],
    cardinality = Total[ Map[ #[[2]]&, NewDeployment ] ]},
  minprice / ( Cost[ NewDeployment ]/ cardinality )
  ];
```

Utility dimension: response time

The first observation is that adding more application servers will decrease the average response time, simply because the measured total response time is then divided by more servers. Just looking at the cardinality of the sets, let | D(k) | be the number of machines in the current deployment, and then for the new deployment candidate of size | D(k+1) | average response time calculated from the measurement will be

$$\overline{T_R}(k+1) \approx |D(k)| * \overline{T_R}(k)) / |D(k+1)|$$

The cardinality of a configuration is just adding together the number of machines ignoring their different types.

```
DeploymentCardinality[ Deployment_List ]/; VectorQ[ Deployment, MatchQ[#,{_,_Integer?Positive}]&] := Total[ Map[ #[[2]]&, Deployment ] ];
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

It should be noted in passing that the deployment cardinality is used because the average response time is measured per deployed machine without taking into account the machine properties. Normally, the more information that can be encoded in the utility function, the more representative it gets. Given that the response time in this use case is dependent on the time it takes to decode a file, the number of cores may be more important than the number of machines. Computing the average response time over the deployed cores could allow less, but more powerful, machines to be deployed.
The next step is then to map this changed response time to a utility. The response time will never be less than zero, and the upper limit is set as T_{max}, hence the interval for the response time is closed. Furthermore, it must be noted that there must be a desired nominal value for the response time. The reason is simply that one may always over-provision application servers, and push the response time down to the level of what would be experienced by one single user per application server. This contradicts the requirement to minimise cost. Hence, the response time should be controlled around a good value that is acceptable response time, but not the lowest possible to avoid incurring unnecessary cost. Let $T_{nominal}$ be the acceptable value, and then the highest utility will be when the measured average response time is close to this value. The utility will be lower if the response time is better than this value because of the over-provisioning of resources, and the utility will be lower if the measured average response time is larger than the nominal value. The two parameters for the response time calculations are

```
T[nominal]=20;
T[max] = 30;
```

The Beta distribution B(α, β) is normally a good model for a family of functions bounded on the interval [0,1], and the expectation of the distribution is

$$E\{B(\alpha, \beta)\}=\alpha/(\alpha+\beta)$$

In general, one may say that the shape parameter β pushes the distribution to the left and lower values and the parameter α pushes it to the right. Given the mapping of the interval [0,30] ↦ [0,1], the expectation of the distribution should be $T_{nominal}$/Subscript[T, max] = 20/30. This creates a binding between α and β:

$$E\{B(\alpha, \beta)\}=\alpha/(\alpha+\beta)=T_{nominal}/T_{max}$$

$$\alpha =T_{nominal}/T_{max} (\alpha+\beta)$$

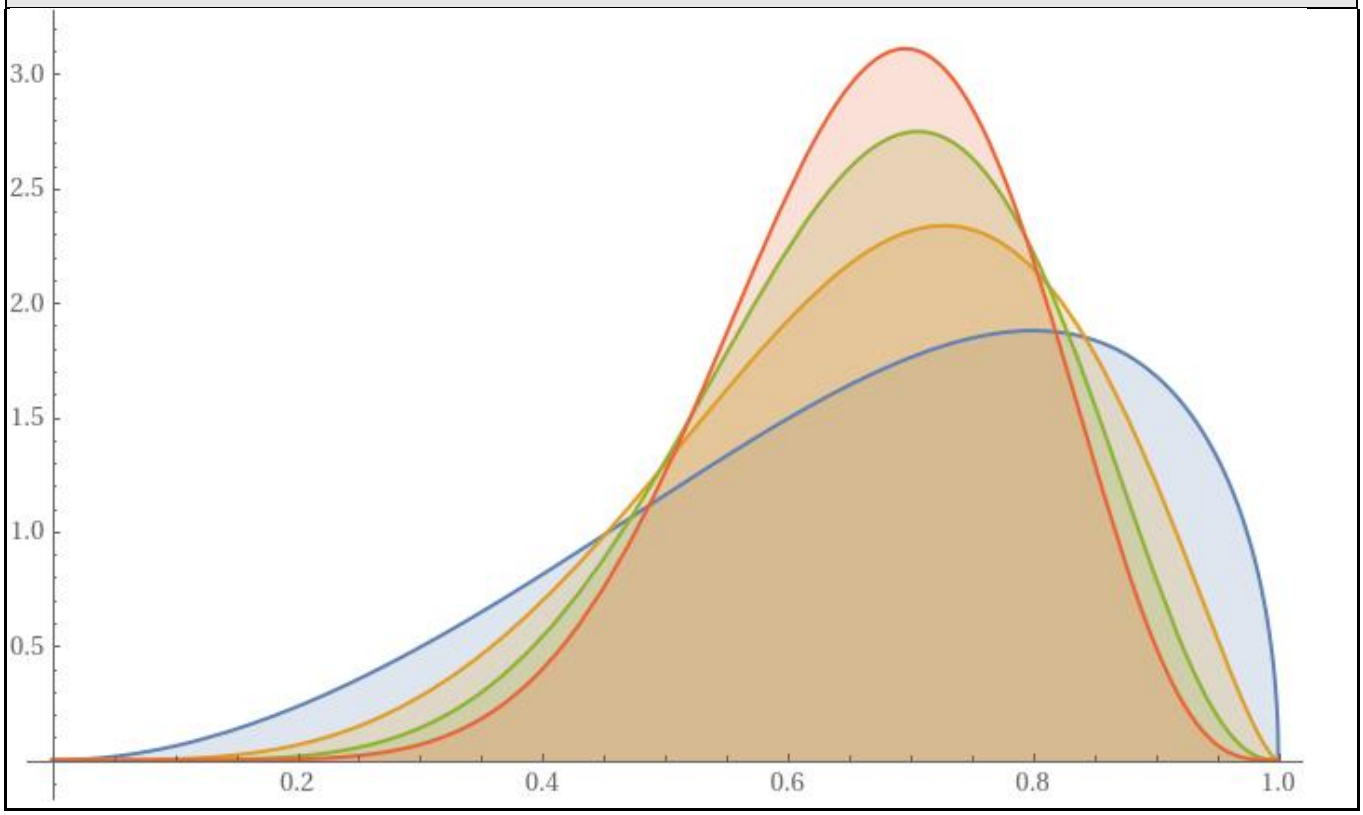
$$(1-T_{nominal}/T_{max})\alpha=T_{nominal}/T_{max} \beta$$

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

$(T_{max}/T_{nominal}-1)\alpha=\beta$

Some example forms based on various values of α is given in the next figure

```
Plot[Table[PDF[BetaDistribution[α,(T[max]/T[nominal]-
1)*α],x],{α,{3,5,7,9}}]//Evaluate,{x,0,1},Filling->Axis,LabelStyle->Directive[FontFamily-
>"Times",Background->White]]
```



Visually, it seems that the blue curve corresponding to α=3 is a good value for this model limiting over provisioning and limiting the utility allocated to response time values higher than the nominal value.

```
α=3;
```

As can be seen from the figure, it needs to be normalised on the maximal value to be a utility in the range [0,1]. For symmetric distributions or high parameter values, the peak value will be located approximately at the expectation, but for skewed distributions, this is not the case, and the peak value has to be computed numerically.

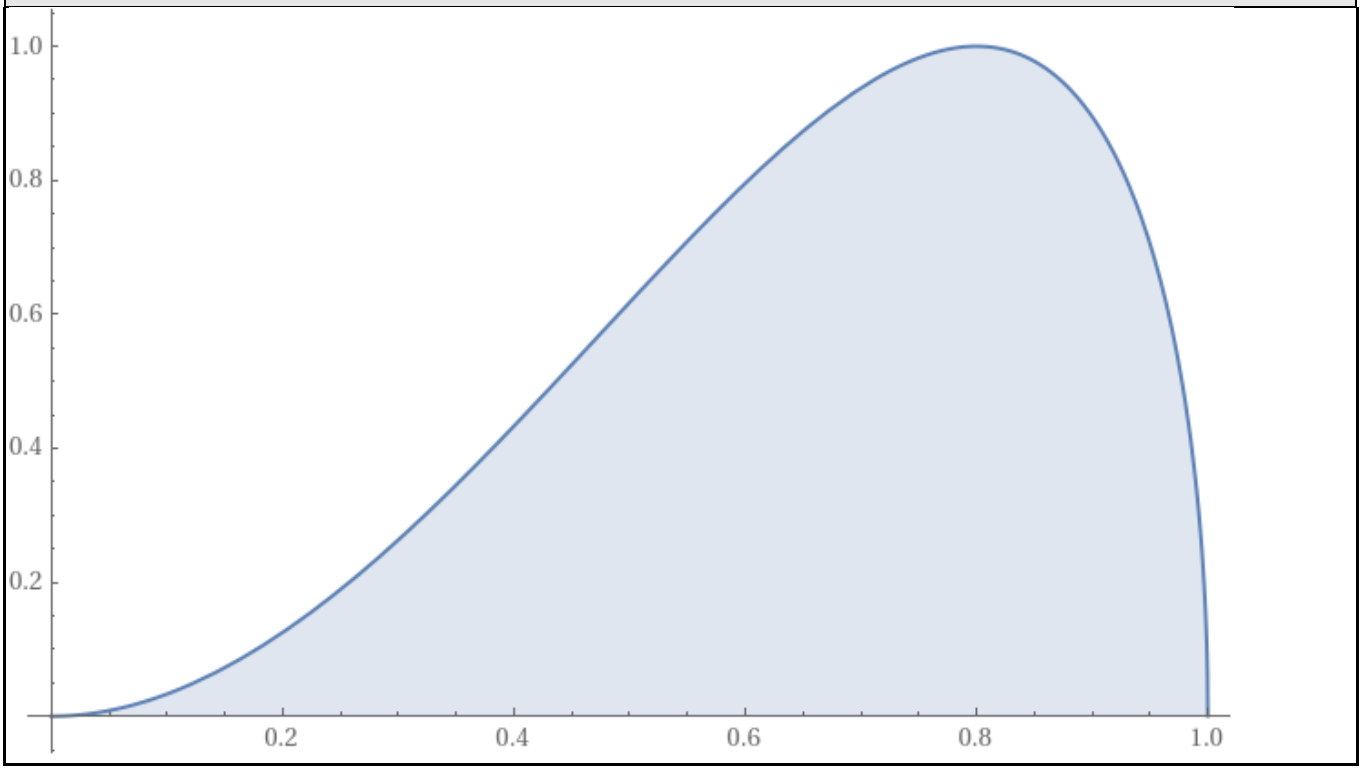
```
BetaNormalisation=  NMaximize[  {N[  PDF[BetaDistribution[α,(T[max]/T[nominal]-
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

```
1)*α],x]],0<=x<=1},x][[1]]
 1.8783
```

The response time utility function is illustrated below

```
 Plot[PDF[BetaDistribution[α,(T[max]/T[nominal]-1)*α],x]/
BetaNormalisation,{x,0,1},Filling->Axis,
LabelStyle->Directive[FontFamily->"Times",Background->White]]
```



The resulting utility based from the change in configuration using Equation (3).

$U_R(k+1|D(k),D(k+1),\overline{T_R}(k))=B(\alpha,\alpha(T_{max}/T_{nominal}-1),\overline{T_R}(k+1)/T_{max}) =$

$B\left(\alpha,\alpha(T_{max}/T_{nominal}-1),((|D(k)|*\overline{T_R}(k))/|D(k+1)|)/T_{max}\right)$

The implementation of this utility function is straightforward.

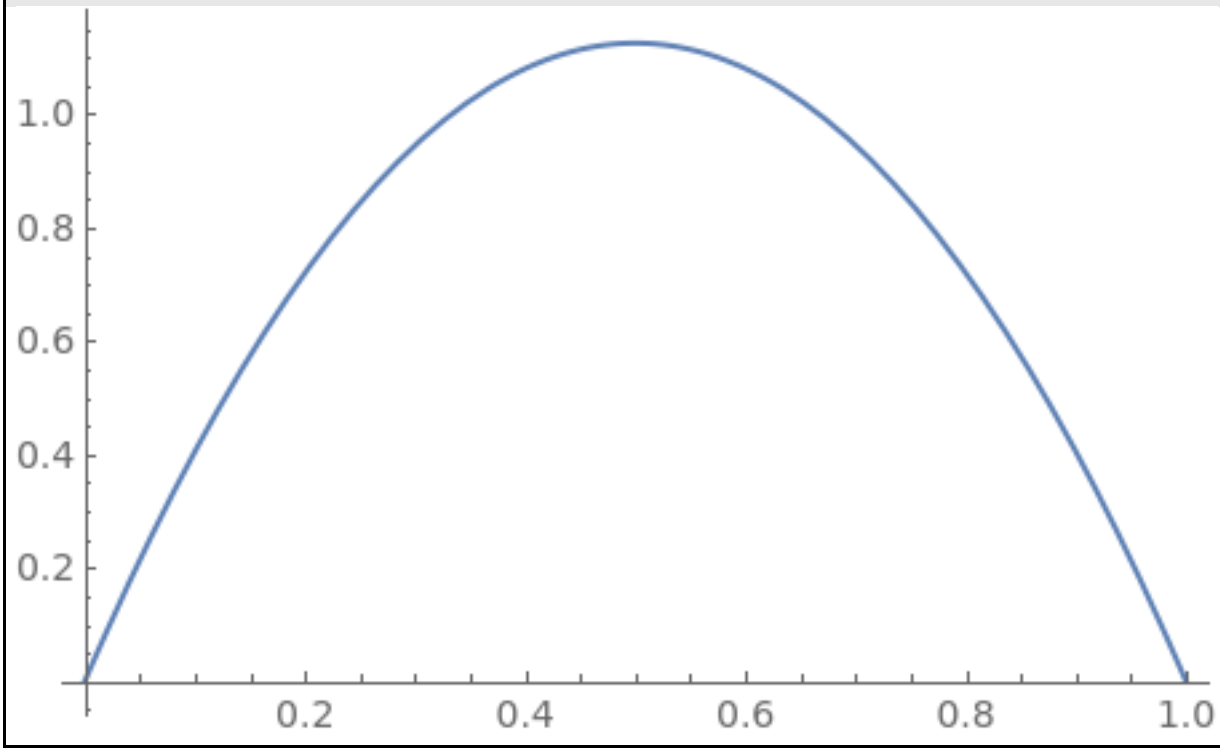
```
 Utility[response, CurrentDeployment_List,
 NewDeployment_List, AverageResponseTime_?NumericQ ] /;
( VectorQ[ CurrentDeployment, MatchQ[#,{_,_Integer?Positive}]&] &&
VectorQ[ NewDeployment, MatchQ[#,{_,_Integer?Positive}]&] &&
TrueQ[ 0<=AverageResponseTime<=T[max] ]) :=
PDF[ BetaDistribution[α,(T[max]/T[nominal]-1)*α], ( ( DeploymentCardinality[ CurrentDeployment
] * AverageResponseTime ) /
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

```
DeploymentCardinality[ NewDeployment ] ) / T[max]  ]/BetaNormalisation;
```

Although this is a possible formulation, it can currently not be supported by CAMEL. One alternative that would be feasible could be to define this utility as a part of slightly rotated ellipsis passing through the three points (0,0), ($T_{nominal}/T_{max}$,1), (1,0) since an ellipse is only defined in terms of squares that can be implemented in CAMEL. One may even achieve something similar with a second order polynomial fit to these points.

```
QuadraticFit = Fit[{{0,0},{T[nominal]/T[max],1},{1,0}},{1,x,x^2},x]
1.1598*10^-16+4.5 x-4.5 x^2
Plot[ QuadraticFit, {x,0,1}]
```



The issue is again that it may overshoot somehow at the maximum value

```
NMaximize[ QuadraticFit, x]
{1.125,{x->0.5}}
```

and it does not mange the non-centrality of the problem.

```
N[ T[nominal]/T[max] ]
0.666667
```

However, it does pass through unity at the nominal response time.

```
QuadraticFit /. {x->T[nominal]/T[max]}
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

```
1.
```

Rather than normalising the quadratic function, it would be better to use the minimum operator and accept that there will be a flat region in the middle.

Overall utility

The overall utility function is then just a normal weighted sum of these two part utilities for a parameter ζ giving the relative weight of the two parts.

$U(k+1|n,v,\overline{T_R}(k))=\zeta*U_{cost}(k+1|n,v)+ (1-\zeta)*U_R(k+1|n,v,\overline{T_R}(k))$

The default value is to place equal importance on the two parts.

```
Options[Utility]={ζ->0.5};
```

The signature of the utility function is basically the same as for the response time utility, with the added option of setting the weight parameter.

```
Utility[SD,CurrentDeployment_List, NewDeployment_List,
AverageResponseTime_?NumericQ, OptionsPattern[] ] /;
( VectorQ[ CurrentDeployment, MatchQ[#,{_,_Integer?Positive}]&] &&
VectorQ[ NewDeployment, MatchQ[#,{_,_Integer?Positive}]&]  &&
TrueQ[ 0<=AverageResponseTime <=T[max] ] ) :=
OptionValue[ζ]*Utility[cost, CurrentDeployment, NewDeployment] + (1-
OptionValue[ζ])*Utility[response, CurrentDeployment, NewDeployment, AverageResponseTime ];
```

Numerical experiment

Assume that there are few users, and the first reading of the response time measurement, indicating few users and a response time of 3 seconds. There are three possible approaches: Increase with one machine, decrease the machine count, or stay with the current configuration:

```
Utility[ SD,InitialDeployment, {{mBig,3}}, 3 ]
0.168575
```

```
Utility[SD, InitialDeployment,{{mBig,1}},3]
0.516573
```

```
Utility[ SD,InitialDeployment,InitialDeployment,3]
0.516573
```

In other words, there is a slight preference to reduce with one the deployed number of machines for this measured response time. Assume instead that there were many users in the system and the response time increased to, say, 25 seconds, then the utility would be.

```
Utility[ SD, InitialDeployment, {{mBig,3}}, 25 ]
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

0.281219
Utility[SD,InitialDeployment,{{mBig,1}},25]
0.995264
Utility[SD,InitialDeployment,InitialDeployment,25]
0.995264

This result may be a little counter-intuitive, but coming from the fact that there is not too large distance between the nominal value of 20s and the measured average response time of 25s. However, if the average response time is measured to 28s, the picture changes.

Utility[SD, InitialDeployment, {{mBig,3}}, 28]
0.307006
Utility[SD,InitialDeployment,{{mBig,1}},28]
0.892919
Utility[SD,InitialDeployment,InitialDeployment,28]
0.892919

If one wants the decision to be made already at measured 25s response time, while keeping the nominal value of the response time at 20s, one could change the emphasis paid to the response time by decreasing the weight of the cost part. This may make good sense in this application where user satisfaction is more important than cost.

Utility[SD, InitialDeployment, {{mBig,3}}, 25 ,ζ->0.05]
0.234317
Utility[InitialDeployment,{{mBig,1}},25, ζ->0.05]
0.991002
Utility[SD,InitialDeployment,InitialDeployment,25, ζ->0.05]
0.991002

This shows that the cost must be de-emphasised significantly for a good decision to be made, and even then the conclusion is not that significant. If 25s is considered a point where a scaling should have been made, then the nominal response time should be decreased to ensure that a scaling decision is taken earlier.
For larger systems the effect of one machine more or less will not change the utility significantly, which is natural given that the cost will almost not change, and therefore emphasising the response time measurement more will help to scale also in this case, but relatively little.

Utility[SD, InitialDeployment, {{mBig,150}}, 28]

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

0.00340076
Utility[SD,{{mBig,150}}, {{mBig,149}}, 28]
0.382397
Utility[SD, {{mBig,150}}, {{mBig,151}}, 28]
0.408619
Utility[SD,{{mBig,150}}, {{mBig,150}}, 28,ζ->0.05]
0.746879
Utility[SD,{{mBig,150}}, {{mBig,149}}, 28, ζ->0.05]
0.720514
Utility[DS,{{mBig,150}}, {{mBig,151}}, 28, ζ->0.05]
0.770416

Example 3: CRM Memory Use

Problem description

The application consists of two *application component types*: The worker computer and a load balancer distributing the incoming users onto the set of workers. There must be at least two worker instances for redundancy, and not more than 6 workers are allowed. The users' activities reflects in increased memory consumption on the worker machines, and the more users allocated to a machine the more memory it needs to serve all the users. The number of users is a slowly changing stochastic process, and one may therefore assume that there will be approximately as many users tomorrow as it was today.

When the memory consumption of a machine reaches 80%, the application will start swapping and the response time seen by the users will be prohibitively slow. Hence, the goal is to provide more workers early enough to prevent this situation of starvation. At the same time, cost is the main concern, and it is desired to use as few and as inexpensive machines as possible. Inexpensive typically implies limited memory, which is fine if the number of users are few. The application utility has therefore been stated as

Minimise deployment cost *while keeping the memory consumption of each machine less than 80%*

The application should consequently *measure* the number of users in total, the number of users per machine, and the absolute and relative memory use per machine.

A machine having live sessions with one or more users cannot be stopped. Hence, without an active collaboration of the load balancer, there is no way to concentrate the allocation of users on the least number of machines. Consider for instance that there has been many users, and 6 worker machines are deployed, but most of these users have logged off and only 5 users remains when a 6th user arrives. Even though each of the 5 machines serving each of the 5 existing users could

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

accommodate the new user allowing one server to be shut down, a perfect load balancer would allocate the new user to the unused machine as it is the best *balance of the load* in the system. Hence, reconfiguring the system by changing the number of worker machines or by changing the deployed machine type to cheaper machines is therefore not possible without the collaboration of the load balancer.
However, the users typically reflect a zero-to-zero pattern in that there are no users of the application at night. This means that it will be possible to reconfigure the deployment of the application for the next day's run based on the measurements obtained from today's run.

Cost utility
As can be seen from the utility statement, the cost is the only thing to minimise for this application. At the same time, there is a bound on the number of worker machines, and therefore consequently a minimum, C_{min}, and maximum cost, C_{max}. A standard linear and normalised cost utility function is therefore proposed for a deployment $D(k)$.

$$U_{CRM}(D(k))=1 - (cost(D(k))-C_{min})/(C_{max}-C_{min})$$

Here $C_{min}$is the cost of using a single machine of the cheapest possible type and $C_{max}$is the cost of using 6 machines of the most expensive type. The cost is computed using the configuration cost of a configuration consisting only of one machine type. The computed cost will be remembered since it is no reason to recompute it at the evaluation of the utility unless the node candidate set has changed.

```
MinCost[CRM,TheNodeCandidates_List?VectorQ ]:=
(MinCost[CRM,TheNodeCandidates] = Min[ Map[ Cost[{{#,1}}]&,TheNodeCandidates] ]);
MaxCost[CRM,TheNodeCandidates_List?VectorQ ]:=
(MaxCost[CRM,TheNodeCandidates] = Max[ Map[ Cost[{{#,6}}]&,TheNodeCandidates] ]);
```

The utility is then a direct implementation of equation (8) for the current set of node candidates given a global list.

```
Utility[ CRM, NewDeployment_List ]/;
( VectorQ[ NewDeployment, MatchQ[#,{_,_Integer?Positive}]&] ) :=
( 1 - (Cost[ NewDeployment ] - MinCost[CRM, NodeCandidates])/
(MaxCost[CRM,NodeCandidates]-MinCost[CRM,NodeCandidates]));
```

Optimisation
It should be evident from the introductory discussion that the optimisation process entails setting the requirement attributes on the *application component types* such that the utility of the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

application is maximised. Here the utility is pure cost. Furthermore, there is one and only one *application component type* to consider: The worker. Based on the attributes assigned for the worker, a different type of *node candidate* will be chosen for deploying the workers, and the cost is a direct result of the choice of the *node candidate* and the number of worker instances.
From the problem description, it follows that the amount of memory provided by the worker machines is the essential quantity. However, neither the amount of users nor their distribution of the workers can be controlled by Melodic. Only the amount of resources provided can be controlled. This leads to the following considerations:
Assuming that the memory requirement attribute of the worker *application component type* is an *interval*, one may take the lower bound of this interval to be be chosen by the solver: RAM_{Low}. The total memory consumed should at the same time be less than 80% of the total amount of RAM provisioned, and consequently one must have that the lower limit of the memory for a worker is less than 80% of the worker's total memory. This leads to the upper bound of the memory for each worker instance to be $RAM_{Upper} = RAM_{Low}/0.8$. It is typically needed to have some security margin since the number of users will change dynamically over the day, and yesterday's maximum number of users may be today's minimum number of users. The discount factor should therefore be a tunable *parameter* $\lambda \in [0, 0.8]$ of the problem. It should be selected based on realistic measurements taken from the running application, and the simulated choices made by Melodic. Based on this reasoning, there are *two variables* of the problem: The lower limit of memory per worker, RAM_{Low}, and the number of worker *Instances*. The memory *requirement attribute* for the worker *application component type* is then given as the interval

$$Memory \in [RAM_{Low}, RAM_{Low}/\lambda]$$

```
Options[Utility]=Join[ Options[Utility],{λ->0.7}];
```

Adding together the absolute maximum memory actually used for each of the current workers, gives the least amount of actual memory RAM_{Total} needed in the system. This lower bound can be provided by one or more worker *instances*. However, since RAM_{Total} is the actual use, and actual use should not exceed a relative fraction of λ of the total provided RAM in the system, one must obviously satisfy the constraint (written to ensure that the left hand side is a constant number and the right hand side contains the variables)

$$RAM_{Total}/\lambda <= RAM_{Low} * Instances$$

It should be noted that the above is assuming the perfect load balancing, so that the memory provided by each worker machine is used more or less to the same degree. If the load balancer is manipulated to pack more users on less machines, then essentially RAM_{Total}/λ may be needed on a single worker machine.
At the same time, it should be an *automatic* constraint set based on the available *node candidates*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

since it could be that there is an upper limit, *i.e.* a constraint on the upper bound of the memory requirement in equation (9). The effect of such a constraint on the problem will then be to force an increase in the number of instances because of the constraint (10). The *domain* of the *Instances* variable is confined to be an integral value in the interval [2, 6], and so there are no constraints necessary to ensure that the number of worker *Instances* stays within the limits of this domain. The actual utility is a function of the assignment to the variables RAM_{Low} and *Instances* for the worker *application component type*, which is the only variables of this problem. The utility function is therefore defined in terms of these two variables. It first sets the RAM_{Low} as a requirement for the CRMWorker component according to equation (9) above, and then uses the pure cost utility to evaluate the utility of a deployment consisting of only the cheapest machine type satisfying the memory requirements. In Melodic, the decision on the two variables' values and the first step will be done by the solver, whereas the machine type selection and the cost evaluation will be done by the utility generator component.

```
Utility[CRM, RAMLow_Integer?Positive, Instances_Integer?Positive, OptionsPattern[] ]:= Block[{},
  SetRequirement[ Memory, CRMWorker, Interval[{ RAMLow, Ceiling[RAMLow / OptionValue[ λ ]]}]
];
  Utility[CRM,{{ SelectCheapest[ SelectNodeCandidates[ CRMWorker, NodeCandidates ] ],
Instances }} ]
  ];
```

When a new reading for the total memory consumption over all deployed machines comes in every night, the Melodic upperware needs to find a new configuration satisfying this *application context*. In practice, the optimisation problem is solved with the memory metric as fixed value. The default upper bound on the memory consumption is defined first, and set equal to the default option for the utility calculation as defined in the optimisation part above.

```
Options[Upperware]^={λ->(λ/.Options[Utility])};
```

Then the optimisation problem is the constrained optimisation problem in the two variables. Note that *instances* is an *application component type attribute* and it cannot be assigned a value by the solver since this will have global precedence and the set requirement function will assign a value to the number of instances, *i.e.* the equation "instances = Undefined" will be understood as "5 = Undefined" and we cannot reassign a number. The solver variable instance is therefore called Cardinality in the upperware implementation.

```
Upperware[ CRM, RAMTotal_Integer?Positive, OptionsPattern[] ]:= NMaximize[
  {Utility[CRM, RAMLow, Cardinality, λ->OptionValue[λ]
],RAMTotal/OptionValue[λ]<=RAMLow*Cardinality && 1<=RAMLow&&
```

```
RAMLow<=Max[Map[(#[Memory])&,NodeCandidates]]*OptionValue[λ]                    &&
2<=Cardinality&&Cardinality<=6  },
  {RAMLow,Cardinality}∈Integers,MaxIterations->1000];
```

Example

To test the above optimisation modelling, the set of machines of the past example is extended, and some memory size is associated with the various machine types. The price model is also extended for the new machines

```
 NodeCandidates = Join[ {mXXS, mXS, mSmall}, NodeCandidates ];
MapThread[(#1[Price]^=#2)&,{{mXXS, mXS, mSmall},{1,3,5}}];
MapThread[(#1[Memory]^=#2)&, {NodeCandidates, {1,4,8,16,32}}];
```

Assume that one fine night the observed memory consumption was 9 GB of RAM, and then the cost optimal *configuration* would be

```
Upperware[CRM, 9]
{0.847458,{RAMLow->8,Cardinality->2}}
```

This follows from the different ways 9 GB can be provided:

- It can be provided by two machines since at least two machines must be used, each with more than 4.5 GB RAM. This can be satisfied by mSmall, which has 8 GB of RAM, or more costly by mBig and mXXL. The cost of this configuration will be

```
Cost[{{mSmall,2}}]
10
```

- Using one machine more reduces the requirement for each machine to 3 GB RAM, and this can be satisfied most cost effectively by the mXS machine having 4 GB of RAM. However, the constraint (10) requires that the total RAM in the deployment must be 9 GB/λ = 9 GB/0.7 = 12.85 GB. Consequently, three machines of 4 GB is excluded because then a too high ratio of memory would be used. The cheapest combination for satisfying this requirement would then be to use three mSmall machines, but then it is obviously cheaper to use only two mSmall machines.
- Given the issue that this application defines one and only one *application component type*, it is not possible to provide different combinations of machines. For instance using two mXS machines and one mSmall machine will provide 2 * 4 GB + 8 GB = 16 GB, which is sufficient. Alternatively, one could provide one mSmall, one mXS, and one mXXS machine, and provide 8 GB + 4 GB + 1 GB = 13 GB. These combinations will cost

```
Cost[{{mXS,2},{mSmall,1}}]
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

11
Cost[{{mXXS,1},{mXS,1},{mSmall,1}}]
9

As can be seen from the above considerations, it would be beneficial to allow inhomogeneous machines for the current price model, but to allow such combinations, different *application component types* must be defined. These should have different requirements for memory, and there should be a constraint saying that the sum of instances over all types should be larger or equal to the minimum two. However, this would also make the problem much harder to solve (and the solver needs more than 100 iterations even for this simple toy example).

Example 4: Simulation: Time to completion

Problem description

In order to generate a training set for machine learning algorithms predicting traffic, it may be necessary to run data farming experiments fed by the data where different scenarios are simulated for a diverse set of parameters. Data farming experiments are typically conducted by an orchestrator that keeps track of which parameter tuples that have been simulated, and makes sure to start new simulations on a pool of worker machines as soon as one worker finishes a simulation job. This is also known as *high throughput computing*. The orchestrator first decides on various parameter sets to simulate, implicitly defining the number of simulations to run. Furthermore, these simulations should all complete within a certain time limit, *deadline*. The utility of the data farming experiment can simplistically be formulated as

*Complete all simulation experiments by the deadline at **minimal cost**.*

Statistical completion time model

It is in general hard to predict how long a simulation run will take because it can depend on the given simulation parameters, and how these affect the data processing, and the data being processed. The simulation run time is therefore a *random variable*. Furthermore, if the workers are running on shared resources, like an organisation's works stations managed by HTCondor or in the Cloud, other activities on the shared hardware will also affect the execution time. Consequently, one cannot predict the duration of a simulation run, but one may *measure* it and use the statistical model to estimate the probable experiment end time based on the amount of resources available.

The *empirical* cumulative density function (CDF) is defined by the measurements. It has a *quantile*, $q_{1-\theta}$, which is a time such that the probability of exceeding this simulation time is given as θ for a parameter $\theta \in [0, 1]$. Let $S(k)$ be the number of simulations remaining at time k when one

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

simulation terminates. If there are *W* worker cores, then one may assume that the simulations will be equally distributed on the worker cores, and that each worker core has *S(k) / W* simulations to do before the experiment finishes. At probability 1 - α these simulations will take a time $S(k)/W\ q_{1-\theta} <= T_{Remaining} = T_{Deadline} - T_{Now}$

Where the upper bound is given by the necessity to complete all simulations by the deadline. Only the number of worker cores, *W,* can be changed by Melodic in order to make the deadline, and thus this is the only *variable* of the problem.
The design parameter θ is a balance between the necessity to finish by the deadline and the cost one is willing to take to make the deadline. Setting θ low means that it is very unlikely that the deadline will not be met, but more worker cores will be necessary. The default is a compromise saying that the deadline may not be met in 5% of the simulation experiments.

```
Options[Upperware]={θ->0.05};
```

The initial estimates for the quantile can either be obtained from historical data, or one has to wait a certain number of simulations before starting to optimise according to the constraint (11).

Cost utility
The cost utility is similar to the *cost bound utility* defined for the secure documents use case above, at the exception that it will use the number of cores as criterion. Hence, the minimal price is given by the machine type that has the least cost per core.

```
MinCost[ Sim, TheNodeCandidates_List?VectorQ ]:= (MinCost[Sim, TheNodeCandidates]=Min[ Map[ (#[Price]/#[Cores])&,TheNodeCandidates ] ]);
```

For a given deployment, the number of cores must be calculated and the average cost per core of the proposed deployment compared with the minimal cost per core in the available set of *node candidates*.

```
Utility[ Sim, NewDeployment_List ]/;( VectorQ[ NewDeployment, MatchQ[#,{_,_Integer?Positive}]&] ) := With[
  { NoCores = Total[ Map[ (#[[1]][Cores])&, NewDeployment ] ] },
  MinCost[ Sim, NodeCandidates ] / ( Cost[ NewDeployment ] / NoCores )
  ];
```

The problem with this approach is that the most core efficient machine will be selected. Always! This means that a machine that has many cores can have a lower price per core than another machine with less cores, although it may be globally cheaper to use many machines with few

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

cores. A very simple and direct cost function is therefore be a better alternative.

```
Utility[ Sim, NewDeployment_List ] /;( VectorQ[ NewDeployment, MatchQ[#,{_,_Integer?Positive}]&] ) := 1/Cost[ NewDeployment ];
```

Optimisation

There must be an automatic constraint to ensure that the number of cores per instance is less or equal to the maximum number of cores of any of the node candidates.

```
MaxCores[ Sim, TheNodeCandidates_List?VectorQ ]:=
(MaxCores[ Sim, TheNodeCandidates ]=
Max[ Map[ (#[Cores])&, TheNodeCandidates ] ]);
```

Since there is only one *application component type* and therefore all instances will be deployed in the same *node candidate* type and the requirements is the minimum number of cores a *node candidate* must have. It will select the cheapest *node candidate* that supports the

```
Utility[ Sim, NumberOfCores_Integer?Positive,
Cardinality_Integer?Positive ] := Block[{},
  SetRequirement[ Cores, SimWorker, NumberOfCores ];
  Utility[ Sim, {{ SelectCheapest[ SelectNodeCandidates[ SimWorker, NodeCandidates ] ], Cardinality }} ]
  ];
```

The execution context changes whenever a simulation run terminates as this will lead to an updated number of simulations remaining, and a new quantile value, both at a reduced time to the deadline. Based on this, the lower bound on the total cores available can be computed from the constraint (11).

```
Upperware[ Sim, SimulationTimes_List,
RemainingSimulations_Integer?NonNegative, RemainingTime_Integer, OptionsPattern[] ] /;
VectorQ[ SimulationTimes, NumericQ ] := With[
  { LowCores = RemainingSimulations * Quantile[ EmpiricalDistribution[ SimulationTimes ], 1-OptionValue[θ] ] / RemainingTime,
   UpperCores = MaxCores[ Sim, NodeCandidates ] },
  NMaximize[ {Utility[ Sim, NumberOfCores, Cardinality],
   LowCores <=(NumberOfCores * Cardinality) && 1<=NumberOfCores && NumberOfCores <= UpperCores && 1 <= Cardinality && Cardinality<=Ceiling[LowCores] }, {NumberOfCores, Cardinality}∈Integers, MaxIterations->1000,Method->"SimulatedAnnealing"]
  ];
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Example

A model for the number of cores on the *node candidates* must be defined in order to solve the optimisation problem. A very simple model is adopted where the number of cores broadly depends on the amount of memory each node candidate has.

```
MapThread[(#1[Cores]^=#2)&, {NodeCandidates, {1,2,4,8,12}}];
```

In addition, there must be a model for the simulation run times. There is no reason to believe that the simulations are dependent. In reality, the parameter sets will be more or less linked, which is an argument for a common expected run time, but the factors affecting the variation is clearly independent of the simulation parameters, and if this is the main cause of the various simulation times. It is therefore reasonable to assume that the execution times will be Poisson distributed around the mean execution time, assumed to be 3 minutes or 180 seconds. 300 samples are drawn for the example.

```
SimExecutionTime = RandomVariate[ PoissonDistribution[ 180 ], 300 ];
```

The quantile can be directly computed

```
q95=Quantile[ PoissonDistribution[ 180 ], 0.95 ]
202
```

If there are 200 simulations to go, it amounts to quite some seconds.

```
200*q95
40400
```

Assuming that the results should be available in one hour, the number of cores needed can be computed

```
200*q95/3600 //N
11.2222
```

The configuration can then be computed with the upperware for this application

```
Upperware[ Sim, SimExecutionTime, 200, 3600 ]
{0.1,{NumberOfCores->12,Cardinality->1}}
```

This confirms the calculation done by hand with respect to the total number of cores. It may be

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

surprising to see that only the largest machine is chosen. Looking at the cost efficiency of the node candidate cores:

```
Map[ N[#[Price]/#[Cores]]&, NodeCandidates ]
{1.,1.5,1.25,0.75,0.833333}
```

Thus there is not a big difference in the cost of cores. However, 12 cores can either be provided by one of the mXXL machines, or two mBig machines, 6 of the mXS machines, and 12 of the mXXS machines, with the respective cost of the configurations

```
Cost[{{mXXL,1}}]
10
Cost[{{mBig,2}}]//N
12.
Cost[{{mXS,6}}]//N
18.
Cost[{{mXXS,12}}]//N
12.
```

Consider then that the problem is the same, but we need the results after 15 minutes. In this case the number of cores needed would be

```
200*q95/(15*60) //N
44.8889
Upperware[ Sim, SimExecutionTime, 200, 15*60 ]
{0.0277778,{NumberOfCores->8,Cardinality->6}}
```

Again, the various alternatives can be investigated by hand. Providing 45 cores will require 4 mXXL machines, 6 mBig, 12 mSmall, 23 mXS, and 45 mXXS. The respective costs of these configurations are

```
MapThread[ Cost[{{#1,#2}}]&,{NodeCandidates,{45, 23, 12, 6, 4}}]
{45,69,60,36,40}
```

Showing that the configuration with 6 mBig machines will be the less costly way to provide this number of cores.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664