HORIZON2020 FRAMEWORK PROGRAMME

TOPIC EUK-03-2016

"Federated Cloud resource brokerage for mobile cloud services"

D6.5
Use cases evaluation and recommendations

Project acronym: BASMATI

Project full title: *Cloud Brokerage Across Borders for Mobile Users and Applications*

Contract no.: 723131

Workpackage:	6	Use Cases Definition, Experimentation and Validation	
Editor:	Sun-Wook Kim		ETRI
Author(s):	Antonia Schwichtenberg		CAS
	Sun-Wook Kim, Song-Woo Sok		ETRI
	Konstantinos Tserpes		ICCS
	Massimo Coppola, Patrizio Dazzi		CNR
Authorized by	Konstantinos Tserpes		ICCS
Doc Ref:	D6.5		
Reviewer	Young-Woo Jung		ETRI
Reviewer	Song-Woo Sok		ETRI
Dissemination Level	Public		

Document History

Version	Date	Changes	Author/Affiliation
v.0.1	23-08-2017	TOC	Antonia Schwichtenberg (CAS)
v0.2	25-09-2017	Add the chapter 2	Sunwook, Kim(ETRI)
v0.3	28-09-2017	Add chapters 3, 4	Antonia Schwichtenberg
v0.4	20-08-2018	Add chapters 5, 7	Antonia Schwichtenberg
v0.5	21-08-2018	Add chapter 6	All partners
v0.6	11-09-2018	Fill chapter 5.1 and 7	Songwoo, Sok(ETRI)
v0.8	12-09-2018	Fill chapters 3.5-3.8, 4.3, 5.3, 8.4	All partners
v0.9	12-09-2018	Internal Review Version	Songwoo, Sok(ETRI)
v1.0	12-09-2018	Final Version	All partners

BASMATI Glossary

Term/Acronym	Definition
Mobile cloud services	Online services offered by cloud resources to support mobile apps. The backend of the mobile apps.
CP	Cloud Provider. The actor that provides the cloud infrastructure/resources, such as VMs
CSP	Cloud Service Provider. The actor that provides cloud services on top of a rent infrastructure from a CP
Cloudlet	Limited capacity infrastructures with virtualization capabilities, often used to support a limited number of users or perform a limited set of operations on behalf of the central cloud infrastructure that hosts the complete application
Edge resources	Resources aimed to operate specialized functionality, located at the "edge" of the network infrastructure, thus, closer to the end users. Examples are (clusters of) Raspberry PIs or cloudlets
BUDaMaF	BASMATI Unified Data Management Framework
KE	Knowledge Extractor
DM	Decision Maker
RB	Resource Broker
MVD	Mobile Virtual Desktop
DAS FEST	An 3-day long music festival taking place in Karlsruhe, Germany every July
ACE	Amenesik Cloud Engine. The cloud service deployment tool through which actual federation is achieved
BEAM	BASMATI Enhanced Application Model. An extension of the TOSCA specification
ASP	Application Service Provider. A Federation user that rents resource services in order to provide an Application services to End-users
Brokering	The matchmaking support provided by BASMATI platform to decide about the best cloud resources to exploit for the execution of the back-end of BASMATI applications. This activity regards the placement of the services or data on computational resources and storages belonging to the cloud data center and

	the cloudlets within the federation.
End user	A user who benefits the various application and infrastructure services provided by the Cloud. Within BASMATI, the most typical example is exploiting the Cloud federation via a mobile device (possibly a laptop) using specialized apps or a web browser.
Federated user	A Federated User is any user whose identity is granted and recognized by the BASMATI identity services. These in turn may rely on identity providers federated within the BASMATI platform, but technical implications are not pertinent here. For what concerns BASMATI, a user whose identity is not tied to any BASMATI recognized identity service can be regarded as an Anonymous user.
Cloud user	The Cloud User is the owner of an application, that is, he/she controls an application that provides a service at the PaaS or SaaS level to a set of end-users. A Cloud user is a federated user. A Cloud user exploits the BASMATI platform to run its application service backend on IaaS resources from the federation. The service provider will use one or more of the cloud providers within the BASMATI federation. The service provider submits a description of its application (intended as a collection of services, with their functional specification) and a QoS that elaborate the non-functional specifications of the application.
Offloading	The ability of BASMATI platform supporting the runtime placement of the components composing the front-end of BASMATI applications on edge resources available nearby the end user. This activity takes place both when edge and mobiles exchange one each other their own workload or when such devices transfer some workload to the clouds or cloudlets. In BASMATI we often distinguish Front-end offloading, related to the mobile part of application, from Back-end offloading, concerning the server side of applications. The latter roughly translates to the known concept of Cloudbursting.
QoE	Quality of experience. It is a measure of a customer's experiences with a service. It may be related to some aspects of the QoS and QoP but can also take into account other metrics.
Service handover	Service handover refers to the activity of transferring an active service between two computational resources (e.g. Cloudlets) with minimal or no disruption on the availability of the service. Ideally, service handover is transparent with respect to the user.
Situational Awareness	The ability of the BASMATI platform to recognise the “situation” characterising the actual combined status of users, applications and resources, aimed at achieving an effective and efficient management of applications and resources.

Executive Summary

In this document the evaluation methodology is described that will be applied in the BASMATI project in order to assess the quality of the BASMATI framework and its components. The evaluation execution is performed by running three use case demonstrators on a BASMATI, cloud federation platform. The results of the evaluation will be presented in this report.

Table of Contents

1 Introduction

The main evaluation purpose in the frame of the BASMATI project is to assess the quality in terms of consistency, correctness and completeness of the components under development. The experience gained by running the three use case demonstrators on a BASMATI cloud federation will be reported and given as feedback to the architecture and interface specification as well as to the concrete BASMATI components implementation. The second and final version of this report will contain evaluation results and will serve as best practices and lessons learned.

1.1 Relation to Other Deliverables

The D6.5 report will provide the evaluation and recommendations of the use cases which integrate the outcomes of the research and development work packages of BASMATI. Considering the evaluation of the use cases with the BASMATI platform defined in the global architecture design of BASMATI (deliverable D2.3), this report will take into account the following as shown in Figure 1.: (i) Use cases Scenarios Definition and Design in the deliverable D6.1; (ii) the role of analysis and modelling of users and applications in the deliverable D3.1;

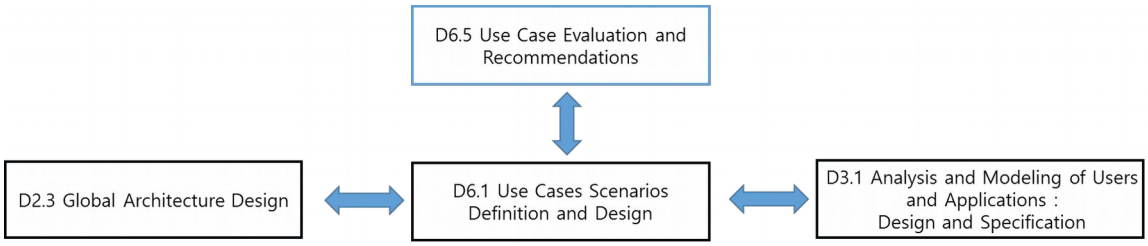

Figure 1 The relationship of D6.5 report to other deliverables

1.2 Outline of Deliverable

Before presenting the evaluation methodology and evaluation criteria that we rely on, we shall start with an analysis and presentation of the components and artefacts that are to be evaluated in BASMATI. In chapter 33 we present the evaluation methodology that we will apply as well as a selection of the evaluation criteria that are relevant for the artefacts under evaluation. For each criterion, dedicated evaluation methods are defined (chapter 3.3) that was applied in the final evaluation execution. In this document we present an implementation and evaluation plan for each use case prototype; the results of the evaluation are reported in chapter 5.

2 What to Measure

2.1 Overview

The evaluation strategy in the BASMATI project has three main goals:

1. To assess the quality and usefulness of the complete BASMATI framework as a result of the use of BASMATI in three different, real life, industrial use cases
2. To assess the achievement of the BASMATI objectives as they have been defined in the description of work
3. To assess the quality of each single component that contributes to the achievement of the BASMATI objectives

To assess the quality in terms of consistency, accuracy and completeness of BASMATI's research and development work package results, three use cases verify the functionality and performance in the cloud federation environment provided by BASMATI. The three use cases should focus on the following items for functional verification of BASMATI.

- Bringing the results of the development work packages into a real life context.
- Showing the feasibility of the main BASMATI innovations and perform a qualitative and quantitative validation of their realization.
- Identifying and designing a set of real-life scenarios where the BASMATI propositions can be exploited in a real-world environment.
- Implementing and experimenting with a demonstrator for each of the scenarios.
- Testing the functionality, effectiveness and quality of the developed technologies.
- Supporting and enriching the BASMATI dissemination and exploitation activities with the developed demonstrators.

Because BASMATI can support a cross-border environment as a cloud federation exploiting worldwide resources and prove not only the robustness of the proposed solutions but also the interoperability between different providers and services achieved, the use cases can show the real life operation and validation such as in the context of large events like DAS FEST, enabling high citizen visibility of the BASMATI solution and potential follow-up interest in EC-funded research. Furthermore, they include the cross-border concept in the case of the Mobile Virtual Desktop UC, a very tangible and practical problem that is encountered in everyday life in the current globalized working environment, affecting a considerable number of citizens and online applications. Finally, the TripBuilder use case is an ideal example of cross-border mobility, challenging the localized resources in terms of performance and privacy/security issues, that affects a highly profitable business sector such as tourist services.

The three use cases are different types of services and require different functions and features for the BASMATI platform. The functions and characteristics required by each use case are as follows:

- Mobile Virtual Desktop
  - Exploits the federated Cloud environment of BASMATI to enhance the QoS associated with the VDI services.
  - Exploits the cross-border brokerage ability of BASMATI to relocate the VDI services from Korea to EU, by exploring the tradeoff cost/performance

- Large Events Use Case
  - BASMATI takes care of QoS of the services provided by means of the advanced offloading and application reconfiguration techniques.
  - Exploits BASMATI advanced tools for user modelling and situational knowledge acquisition for deciding in a proactive way where and when allocate computational and storage resources.
- TripBuilder Use Case
  - BASMATI Cloud federation can help in relocating the services to different cloud providers.
  - D2D capabilities of BASMATI can orchestrate the offloading process to the cloud infrastructure

The three use cases, for assessing quality in terms of consistency, accuracy and completeness of BASMATI's research and development work package results, focus on each of the functional requirements outlined above. A dedicated evaluation framework for that goal has been defined in chapter 3, the results are summed up in chapter 5 – detailed interview results are presented in the appendix chapter 8.

For the second and third goal: To evaluate the achievement of the BASMATI objectives and to measure if and how the single components are contributing to these, we performed a component level evaluation as described in chapter 6.

2.2 BASMATI Components

In this section, we give an overview of the software components that have been developed in the frame of the BASMATI project and that are evaluated in this document. As described in the previous section, all single components are evaluated especially with respect to their contribution to the project´s objectives and the added value:

- BEAM Document processor (chapter 6.8)
- Application Repository in Application Back-end Management
- Knowledge Extractor in Application Back-end Management (chapter 6.6)
- Decision Maker in Application Back-end Management (chapter 6.11)
- Application controller (chapter 6.1)
- Resource Broker in Federation Management (chapter 6.2)
- EDGE Resource Management (chapter 6.9)
- Cloud Provider Management (chapter 6.10)
- Cloud Federation controller (chapter 6.7)
- Federation Data Management in Federation Management (chapter 6.5)
- Federation monitoring (chapter 6.3)
- Federation SLA Manager (chapter 6.12)

Considering the interaction of the use cases with the BASMATI platform as defined in the global architecture design of BASMATI, the use-case based evaluation and recommendations will take into account not only, but especially the following components as shown in Figure 2:

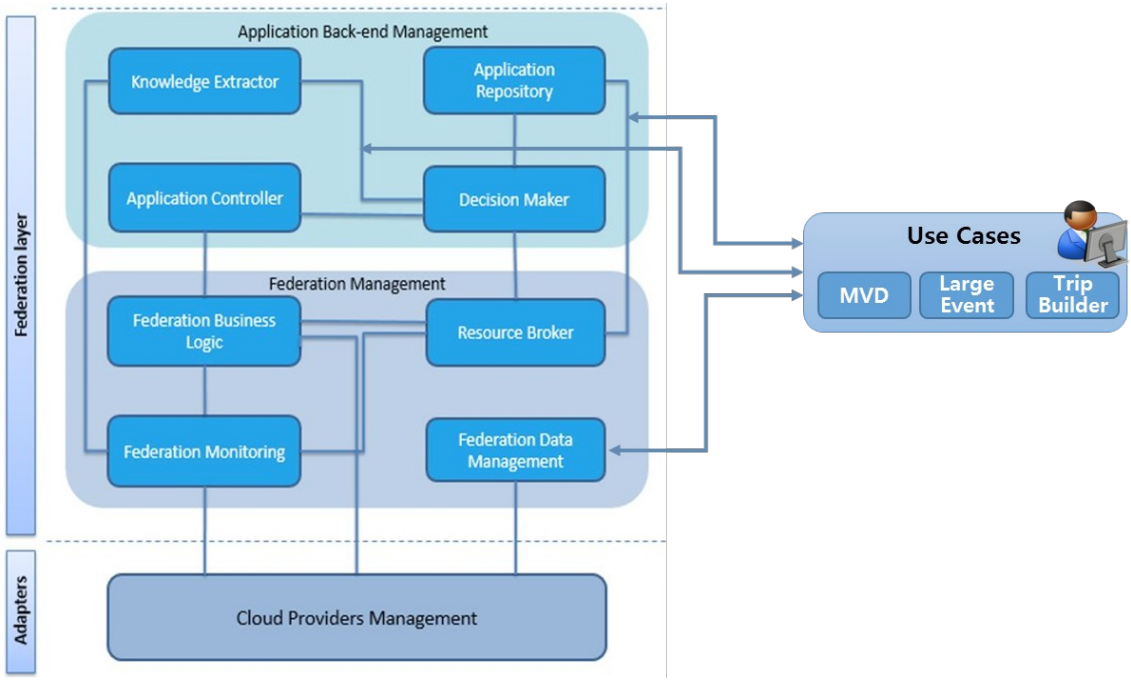

Figure 2 The relationship of use cases evaluation to other BASMATI Components

As you can see from Figure 2, the three use cases are closely related to the Application Back-end Management and Federation Management. The main component in BASMATI related to the Application Back-end Management is the Knowledge Extractor. **The Knowledge Extractor** (KE) is the BASMATI platform component that produces useful predictions about the users' behaviors, the application pattern usages and the sessions between users and applications.

Also, the main components in BASMATI related to the Application Back-end Management are the **Decision Maker** which is entitled of taking decisions about application placement, taking into consideration requirements and previsions as well as all the available information relating to the footprint generated by applications regarding resources during their past executions. The **Application Repository** (AR) stores the high-level information relative to applications and provides the interfaces to access them (write, read, delete). By its role, the design of the AR is strongly interlaced with the concepts expressed by the BEAM representation of the application.

The main components in BASMATI related to the Federation Management are the Resource Broker and Federation Data Management. The **Resource Broker** is responsible for providing the tools and mechanisms for the decision about the placement of, or parts of, the application on

federated resources. The BASMATI Unified Data Management Framework (BUDaMaF) in **Federation Data Management** components will employ a set of different connection methods with other components of the BASMATI framework. The application data in the Data Management Framework is the data used by applications allowing them to provide their services.

As the demands of the use cases are different, the BASMATI framework was instantiated slightly differently to support the deployment and federation needs of each of the use cases. Chapter 4 gives an overview of the components that are used in the use case demonstrators. More detailed information on the demonstrator´s architecture can be found in the documentation of the use case demonstrators (Deliverables D6.2, D6.3 and D6.4).

3 How to Measure

3.1 Evaluation Methodology

After the analysis of different evaluation methodologies and frameworks [CITATION Mar03 \l 1031], [CITATION Jos14 \l 1031],[CITATION Ste02 \l 1031] and[CITATION Tim06 \l 1031], we decided to rely on the so called Tailored Quality models [CITATION Jos14 \l 1031]. Tailored Quality Models represents a conclusion of the insight that a common and general evaluation method cannot be applied to software evaluation because the software and the involved stakeholders are to diverse [CITATION Dro95 \l 1031]. A Tailored Quality Model starts with a set of high-level quality criteria that are broken down to concrete criteria and metrics that are relevant and valid for the artefacts under evaluation.

3.2 Evaluation Criteria

In BASMATI we rely on the ISO standard 25010:2011 [CITATION ISO11 \l 1031] which defines the following set of criteria:

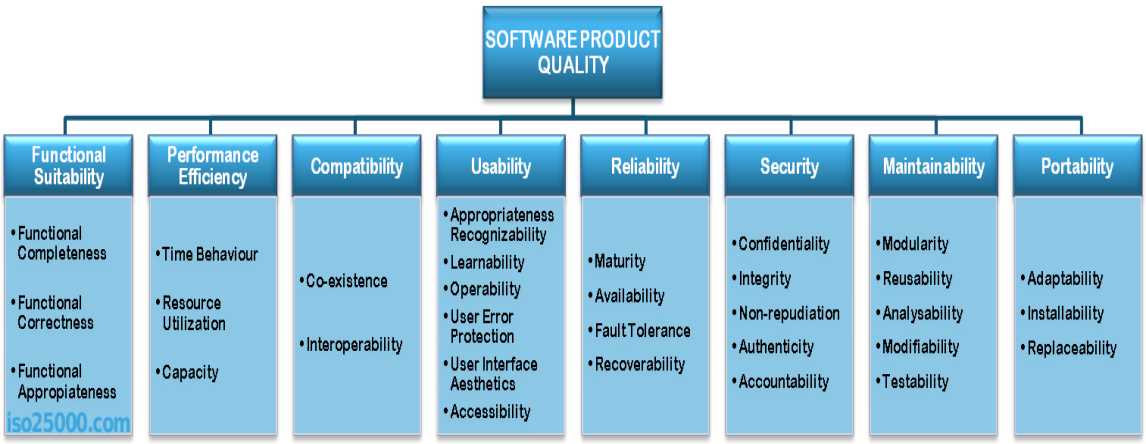

Figure 3: ISO 25010:2011Software evaluation criteria

This set of criteria defined by ISO 25010 has been adapted to be usable to measure the quality of the BASMATI components. Figure 4: Evaluation Criteria and Evaluation Methods gives an overview of the set of criteria and the assigned measuring method. The following chapter 3.3 and 3.4 describe in more detail the evaluation methods indicated here.

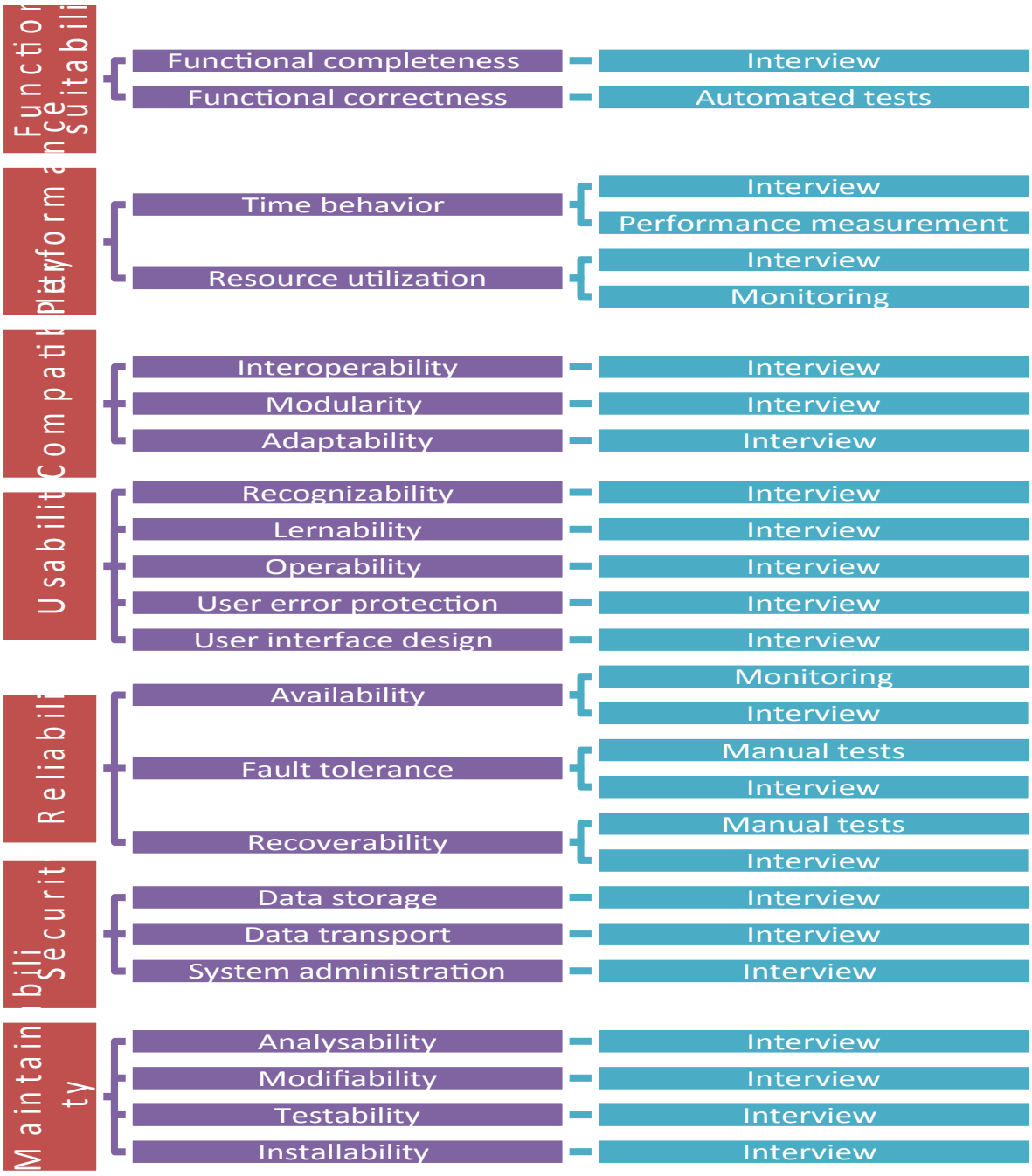

Figure 4: Evaluation Criteria and Evaluation Methods

3.3 Evaluation Methods and Metrics

In BASMATI we use the following evaluation methods:

- Interviews

- Automated tests
- Performance measurements
- Monitoring
- Manual Tests

Based on background information about the artefacts that will be evaluated (chapter 2), we create dedicated interview questions, performance and monitoring metrics as well as an approach towards automated and manual testing depicted in the following sections. For the most relevant criteria, we also performed a comparison between the deployment and runtime of the use case application without and with BASMATI.

3.4 Questionnaires and Interview guidelines

In order to assess the quality of the BASMATI components in terms of the defined criteria, the questions below are defined. In order to get valuable feedback, also about possible improvements, these questions will in most cases be asked in dedicated interviews instead of form-based questionnaires. For the person conducting the interview it is essential to ask the interviewee for constructive feedback.

3.4.1 Stakeholders and Roles

For the interview to be as representative and valuable as possible, it is important to find the right people to be interviewed. As defined in D2.3 Global Architecture Design as well as in the glossary, we consider the following user roles for the BASMATI framework:

- Cloud User
  In BASMATI we consider Cloud Users as persons owning or operating a cloud application. Therefore, a BASMATI Cloud user may fulfil the role of an application developer and/or a cloud application operator and/or a business manager of a cloud application that is deployed and operated with the help of BASMATI.
- End-User
  In contrast to a cloud user, an end-user in BASMATI is considered as a person that uses the cloud application that is deployed and operated by BASMATI. An end-user normally does not interact with BASMATI – only in the case where the end-user also owns services managed by BASMATI.

The interview has been performed with persons fulfilling the role of a cloud user as defined above, especially with application developers and application operators. The feedback of the use case partner´s business managers instead is rather reflected in D7.3 Exploitation plan. In the DAS FEST and in the trip builder use case, the end-user of the application is not a BASMATI user in the sense that the end-user is neither known to BASMATI nor interacting directly with BASMATI components. Therefore, only in the Mobile Virtual Desktop use case end-users are interviewed.

Table 1 below encompasses all questions that can be posed within the interviews. For each use case, the conducted interviews are summed up in chapter 5. Original questionnaires with answers are provided in chapter 8.

Criteria	Attribute	Questions
Functinal suitabiliy	Completeness	Does the system adequately cover the needs of the application? Does it accommodate the platform requirements of the application (e.g. operating system, Java environment, etc.)?
Performance	Time behavior	How long does it take to deploy a normal mid-size application (e.g. having 2 comonents running on two VMs)? Are scale-in and scale-out events handled in time so that a good time behavior of the deployed application is guaranteed?
	Resource utilization	How much hardware (harddisk space, RAM, CPU) are needed for the whole BASMATI framework itself to run?
Compatibility	Interoperability	Does BASMATI support the deployment of any application with any OS and hardware requirements? Which restrictions are already known?
	Modularity	Can different modules of BASMATI be used in isolation?
	Adaptability	Can single modules of BASMATI be adapted? Are extension points provided to do so? Is the source code well-documented or self-explanatory?
Usability	Recognisability	Do the component interfaces follow standard patterns?
	Learnability	Are the interfaces (UI and API) laid out logically, allowing the system to be learned quickly?
	Error Protection	Does the system prevent incorrect parameter values from being specified? Does the system timely identify any incompatible configurations that would prevent the correct deployment/operation of the cloud application?
	User interface design	Is the user interface pleasing? Is it well designed?
Reliability	Availability	Please report on the conducted monitoring.
	Fault tolerance	Did you experience downtime of the platform during your tests? If yes how many times? How many errors were raised without affecting your experience with the platform? If yes, please describe the errors. How many fatal errors were raised?
	Recoverability	Please report on the mean time to recover BASMATI as well as to recover your application (components).
Security	Data storage	Is it possible to access the application data? What kind

		of security mechanisms are in place in order to secure your applications data? Can your own custom security mechanisms be use to secure our application?
	Datatransport	Which mechanisms are in place to secure the data transport (a) between your application components and (b) between your components and BASMATI?
	System administration	Which mechanisms are in place to securely access (a) BASMATI provisioned VMs and (b) the BASMATI administration UI?
Maintainability	Analysability	Can errors and bugs easily be traced? Are sufficient logs provided?
	Modifiability	Is the source code available for everyone? Can bugfixes be committed by everyone? If the answer is "no": Who is in charge of doing so?
	Testability	Can the whole BASMATI framework be tested or single components? Is it clear how to test the components? Please report on the test coverage if possible.
Overall	Ease of Use	Are the components easy to install, configure, and use?
	Usefulness	Were you able to operate your application with the BASMATI services? Exlain the main benefit of using BASMATI.
	Documentation	Is a documentation of all BASMATI components the user needs to interact with available? Do you consider the documentation useful?

Table 1: Evaluation Criteria and Questions for the Assessment

3.5 Performance Tests

Performance tests were conducted for many of the BAMSATI components. The performance evaluation metrics are defined for each such component in detail in Section 6 and are summarized here:

Metric	Components	Evaluation Target	Description
Deployment Time	Application Controller	< 200 seconds	Deployment time of a hosted application on federation resources.
Scaling Time	Application Controller	< 150 seconds	The time needed to perform scaling tasks on a hosted application.
Candidate List Generation time	Resource Broker	< 10 seconds	The time needed to generate a list of

			best-fit candidate plans.
Placement Plan Generation time	Decision Maker	< 60 seconds	The time needed to generate and provide a placement plan.
Scaling Response Time	Data Management Framework	<= 180 seconds	The time needed to perform scaling tasks on a data store.
Average % of resource usage per 10 minutes	Data Management Framework	> 90%	The average percentage of used resources on all data servers in 10 minute windows.
% of requests out of specified time	Data Management Framework	< 1%	The percentage of SLA timeout infractions for a pre-specified time window.
Prediction Accuracy	Knowledge Extractor	> 87%	The prediction accuracy of the model that estimates the resources needed by a hosted application.

3.6 Monitoring Metrics

For many of the BASMATI components, there was a metric the value of which was indicative of the achievement of the component's primary objectives. These are presented in detail in the table below.

Components	**Monitoring Metric**	**Evaluation objective**
Application Controller	Deployed federated resources	Justified (e.g. with financial criteria) selection of deployed federated resources
Resource Broker	Available federated resources	Completeness in considering resources of all members of the federation
Decision Maker	SLA Violations	The resource-application requirements mapping should at least lead to a minimum penalty
Data Management	Exposure of data in its	Privacy leakage concerns

Framework	original form	should be minimized
Knowledge Extractor	SLA Violations	Predicted resource needs should meet the application requirements

3.7 Automated Tests

Automated testing is best suited to the following areas/scenarios[1]:

- Regression Testing: Here, automated testing is suitable because of frequent code changes and the ability to run the regressions in a timely manner.
- Load Testing: Automated testing is also the best way to complete the testing efficiently when it comes to load testing.
- Repeated Execution: Testing which requires the repeated execution of a task is best automated.
- Performance Testing: Similarly, testing which requires the simulation of thousands of concurrent users requires automation.

The majority of BASMATI components (with the exception of the BEAM Document Editor) underwent the procedure of automated tests.

3.8 Manual Test

Manual testing is best suited to the following areas/scenarios:

- Exploratory Testing: This type of testing requires the tester's knowledge, experience, analytical/logical skills, creativity, and intuition. The test is characterized here by poorly written specification documentation, and/or a short time for execution. We need the human skills to execute the testing process in this scenario.
- Usability Testing: This is an area in which you need to measure how user-friendly, efficient, or convenient the software or product is for the end users. Here, human observation is the most important factor, so a manual approach is preferable.
- Ad-hoc Testing: In this scenario, there is no specific approach. It is a totally unplanned method of testing where the understanding and insight of the tester is the only important factor.

In the case of BASMATI, manual testing was only used in the case of the BEAM Document Editor.

[1] https://dzone.com/articles/automated-testing-vs-manual-testing

4 Evaluation Execution Plan

As planned in BASMATI, the software prototypes were released in their final version in month 22 (March 2018) and the integration is done at month 24. (May 2018). As dictated by the integration strategy, the API and communication design were decided early in BASMATI, the components were already integrated and could be used as such in the use cases implementations.

During the implementation of the use case, the three use case partners had gathered experience with setting up and using the BASMATI framework. Feature requests and feedback during that process have been communicated back directly to the BASMATI partners that were responsible for the BASMATI component in question. The overall evaluation of the final BASMATI framework and its components had been performed during the final months of the project.

4.1 Mobile Virtual Desktop Use Case

The Mobile Virtual Desktop (MVD) use case has been evaluated on the BASMATI platform from the start of May to the end of July 2018.

In the MVD use case, the following BASMATI components have been used and evaluated:

- Decision Maker in Application Back-end Management
- Business Logic in Application Back-end Management
- Application Repository in Application Back-end Management
- Application Controller in Application Back-end Management
- Resource Broker in Federation Management
- Cloud Management Platform in Federation Management
- Federation Monitoring in Federation Management

The MVD application is composed of 1 Connection Manager and multiple HOSTs. The Connection Manager is the gateway for MVD users. The HOSTs host users' virtual desktop VMs. To enhance the user experience of MVD, the latency should be minimised.

If a user journeys to distant country or continent, the user will experience very slow response time caused by the physical distance and latency. To mitigate slow response time, the user's VMs should be moved to new HOSTs that are located near to the user's destination. In order to do that, the MVD application should operate with HOSTs that are located at different regions or continents.

Following Figure 5 shows the MVD application's BEAM that is configured with multiple HOSTs located at different regions or countries. To support scalability of HOSTs, the HOST section

includes Scalability Engine features that make the number of HOSTs variable in alignment with the request rate.

The MVD use case shows the following features of the BASMATI platform:

1. Support for Multi-provider, Multi-region applications using Cloud federation.
2. Scalability of application based on monitored resource usage.
3. Optimisation using multiple requirements – placement, VM specification with special feature (host-passthrough), price, performance, SLA, and more.

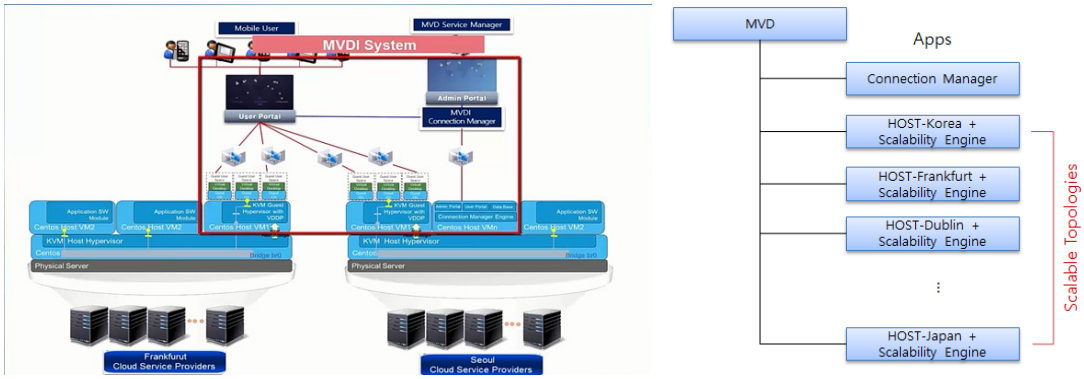

Figure 5: Architecture of MVD

4.2 Large Events Use Case

As CAS/YM´s final use case is the DAS FEST taking place from 19th to 21st of July 2018, the result of the evaluation will be reported in an update of this document directly after the festival takes place.

The following **Figure 2** depicts the bird´s eye view of the components involved in the realization of the real-time demonstrator that has been set up for the DAS FEST 2018.

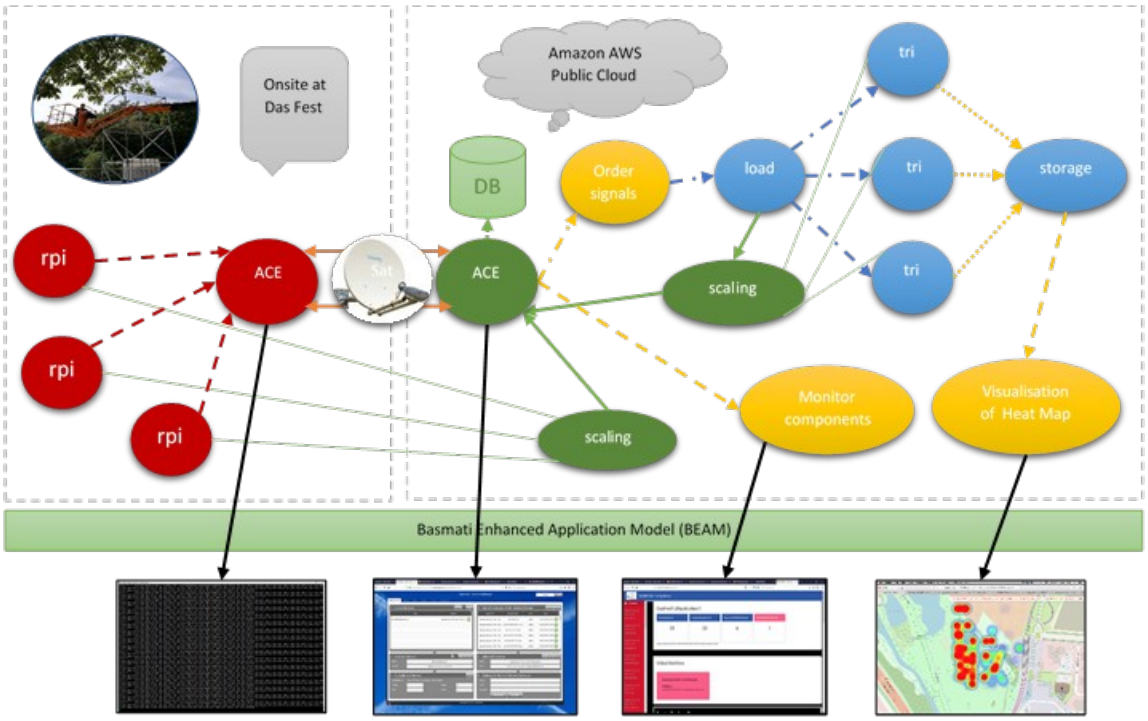

Figure 6: Components of the DAS FEST demonstrator 2018

The Large Events Use Case Application presents an almost real-time Heat Map view of the calculated geo-location of all mobile phone carrying visitors to the DAS FEST event. Mobile phones are passively and anonymously detected by a network of raspberry Pi devices positioned at strategic positions around the event area. They periodically deliver their data to the onsite edge device controller which in turn delivers the data over the satellite link up to the application controller in the cloud. The signal ordering component sorts and filters this data which is then presented to the load balancer for distribution over the scalable collection of trilateration components responsible for the calculation and storage of the geo-location information driving the heat map visualisation view.

This use case demonstrates three important and innovative features that have been designed and implemented during the Basmati project. Firstly, the **management of the provisioning and allocation of the raspberry Pi edge devices** is performed by a generic edge device provisioning extension to the standard Amenesik Cloud Engine, ACE. Secondly, the Amenesik Cloud Engine responsible for the management of the application and its cloud resources, makes use of the **dynamic cloud federation technology** of Basmati allowing the edge provisioning platform to make onsite edge device resources available for use by the **cloud application controller**. Finally, the **Basmati Enhanced Application Model, BEAM**, based on the international standard TOSCA, processed by the Application Controller extension to ACE, is used for the management of the description, **deployment and life cycle control of all application components both onsite and in**

the cloud. More details about the DAS FEST use case and on the implementation details of the components involved can be found in deliverable D6.3 Large Events Use Case.

4.3 TripBuilder Use Case

TripBuilder is a system helping tourists to build their own personalized sightseeing tour. Given a target touristic area, the time available for the visit, and the tourist's profile (a set of preferences rating different categories of interest), TripBuilder provides its users with a time-budgeted tour that maximizes tourist's interests and takes into account both the time needed to enjoy the attractions and the time to move from one Point of Interest to the next one.

Practically, the TripBuilder application for BASMATI consists of an online server that can be decomposed into a web front-end, a backend knowledge base and a module solving the combinatorial problem implied by each user query. The query computational latency is significant and critical, as it is part of the user perceived delay and thus directly impacts the quality of experience.

The TripBuilder evaluation has been performed at various phases during the last year of the project. The evaluation has been organized in four main phases:

1) **Functional Evaluation**. This evaluation phase is concerned with checking if the TripBuilder software launches and behaves correctly from a functional point of view when running with the BASMATI platform. The evaluation consisted of preparing a BEAM version of the TripBuilder application, load it in the BASMATI platform and run it from there. This phase has been conducted by using several components of BASMATI instanced elsewhere. The components that have been used in this phase are:
    - Application Repository
    - ACE engine

2) **Quality of Experience**. The aim of this phase is to assess quality of experience parameters as well as the relative monitoring metrics for the TripBuilder application. This phase consisted of sending an increasing number of requests (with different parameters) to a TripBuilder instance and measuring its response time. This phase has been carried out at the CNR premises and no BASMATI components were involved. The results of this phase instructed the subsequent evaluation phases.

3) **Multiple Instances.** This evaluation phase's aim is to assess whether the BASMATI platform correctly support the deployment of a TripBuilder application consisting of multiple frontends, in fact having multiple access points running on different geo-located cloud providers. This phase required an improved BEAM version of the TripBuilder application. It involved the following BASMATI components:

- Application Repository
- ACE engine
- Decision Maker
- Knowledge Extractor
- Resource Broker
- Application Controller

4) **Load Balancing.** Finally, the last evaluation phases regarded the ability of BASMATI to switch on and off instances of the TripBuilder application according to quality of services violations measured by the monitoring of the application. This evaluation phase involved the following BASMATI components
   - Application Repository
   - ACE engine
   - Decision Maker
   - Knowledge Extractor
   - Resource Broker
   - Application Controller
   - SLA Manager

5 Evaluation Results

5.1 Mobile Virtual Desktop Use Case

5.1.1 Comparing MVD with and without BASMATI

The evaluation of Mobile Virtual Desktop (MVD) has been conducted using the methodology defined in Section 4.1. From a functional standpoint, the Basmati platform has created exactly the required types of virtual machine instances in the federated cloud environment for MVD use cases based on the defined BEAM. It also supported automated installation and configuration of MVD Connection Manager and MVD hosts which requires nested virtualization hardware feature.

When the creation of virtual machine instances, for the MVD hosts in a required region was requested in order to guarantee the QoS for the users move into the area, BASMATI allocated appropriate virtual machine instances which were hosted by cloud providers placed near the service region.

In addition, BASMATI provides a function interface for expanding and reducing virtual machine instances according to the change of service load. So, it enables service providers to efficiently utilize resources without managing separate cloud infrastructures, thus contributing to cost savings.

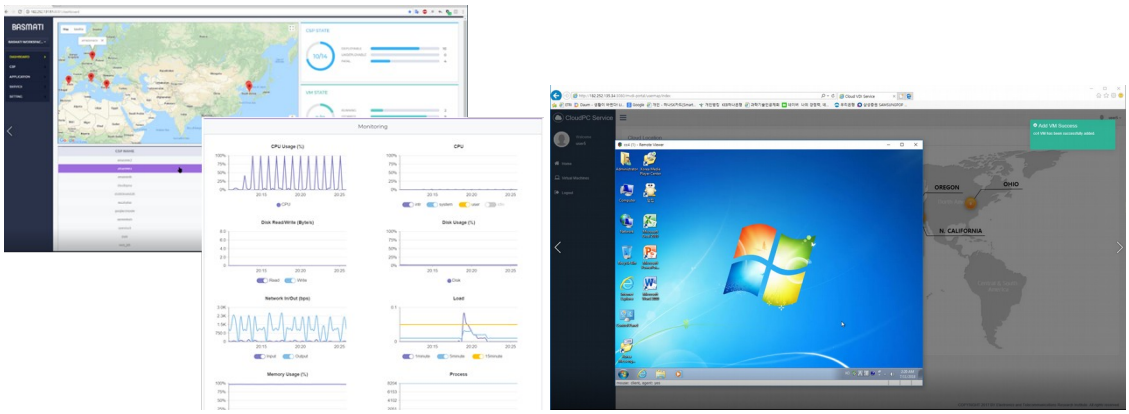

Figure 7. MVD Use Case running on BASMATI Platform

Without BASMATI platform, MVD service provider needs to select the cloud infrastructure that provides the hardware features and directly select the flavour of the virtual machine instances through the appropriate infrastructure for the connection manager and MVD hosts. Then, he/she must install the MVD service him/herself.

In particular, there is a limitation in providing uninterrupted services to users because they do not know the information provided by the cloud infrastructure in the area when moving between regions. In order to manage virtual resources among multiple cloud service providers, it is necessary to control expansion and contraction of virtual instances using a separate infrastructure management tool rather than BASMATI-based MVD Connection Manager. There are inconveniences in using multiple different management tools when using multiple cloud infrastructures.

The following table shows the summarized differences between servicing MVD without BASMATI and with it.

Criteria	MVD without BASMATI	MVD with BASMATI
Resource Utilization	It should use each clouds' available resources statically and manually. It means a lot of resources should be prepared and wasted in order not to degrade user experience. For random patterns of user requests, the resource utilization for static deployment of MVD is about **22~31%**.	BASMATI enables dynamic resource allocation to keep up with changes of users' requests and movements. So, MVD use case can utilize federated cloud resources dynamically and optimize utilization without compromising users' experience. For the same patterns of user requests, MVD using BASMATI platform utilizes **65~68%** of resources.
Costs	Deployment time – it took a lot of time to select proper cloud	At first, it needs some efforts and time to model MVD in BEAM format.

	providers, instance flavours, number of instances, and installation and configuration of services. Resource cost – Because of inflexibility, it needs to use more resources than what it really needs. Or user experience can be degraded if users' requests exceed capacity of deployed resources.	But, since BASMATI can select proper cloud providers, flavours and automate installation and configuration of the service, it can reduce service deployment time: we can deploy the complete MVD service in less than **10 minutes** and scale up a new host in less than **5 minutes**. Thanks to BASMATI platform, MVD can be prepared for a service region in very short time – less than 10 minutes. So, resource cost can be saved a lot than without BASMATI. For same QoS level for users, we can save **45~60%** of costs compared to the "without BASMATI case". The number varies with the change of user request patterns.
QoS - Latency and Response time	User would experience very slow response for the MVD service when traveling across borders. When user who stays in Korea uses MVD running on EU side, the response time can be more than 200ms. – the user can feel far degraded responsiveness. In addition, using the MVD service can be uncomfortable.	BASMATI can select and provide cloud service resources based on the region. Therefore, MVD can deploy new host near the user and move the user's MVD instance to the host. Then, the response time can be less than 10 ms – in our experiment, it was 5.3ms. User can feel great responsiveness for his/her MVD.
Governance	To use multiple different cloud services, it needs multiple different management tools for distributed MVD application. - It is difficult and complicate.	With BASMATI, we need to talk with BASMATI only. It means that we need just one tool to manage whole MVD services deployed on multiple clouds.
Scalability	To scale up and down the instances on multiple clouds, we need to use different tools and APIs. Moreover, some providers don't have scale up/ down features.	BASMATI can support scaling up/down features even if underlying cloud providers don't provide them. Therefore, MVD can increase and decrease the resource instances region by region in real-time.

5.1.2 Interview summary

We have conducted 2 interviews with experts in our project. Generally they have been satisfied with the test results of MVD use case on BASMATI platform.

They experienced some difficulties during preparation of the BEAM documents for MVD use case. Even though they were provided BEAM manual and BASMATI documents, the examples of actual BEAM they can reference were limited in number.

However, there had been no critical issues when running MVD application after the BEAM was composed correctly. The behavior of BASMATI platform was executed as expected and the performance of whole experience was reasonable and acceptable.

Thanks to the multi-region, multi-provider supporting feature of BASMATI, the MVD application could utilise the resources from different openstack-based providers in different-regions to optimise user experience in terms of response time. While the management of the application's deployment was very easy and convenient because it needs only the BEAM to be tackled by the user. A summary of the interview is showed in the following section.

- Functional suitability
    - Functional completeness: All BASMATI functions for MVD services was well provided. When the service provider requests the MVD service, the virtual server and service are automatically deployed to the region or near location by BASMATI.
    - Functional correctness: BASMATI functions for MVD services worked correctly.
- Performance
    - Time behaviour: Transferring the golden image of Virtual Desktop was not an issue for BASMATI. VM allocation and service installation for MVD services had been completed in 5 minutes. This is reasonable and acceptable.
    - Resource utilization: Connection Manager - 8GB RAM, 4 vCores, 50GB disk, Service Platform – 12 vCPU, 16GB RAM, 300GB disk -as usual and fit for the requirements.
- Compatibility
    - Interoperability: No restrictions
- Usability
    - Recognisability: Good
    - Learnability: Some pre-knowledge is needed to use BASMATI and BEAM correctly
    - User error protection: No
    - User interface design: Can be improved
- Reliability
    - Availability: Almost 100% during the pilot service
    - Fault tolerance: Can support using cold migration
    - Recoverability: The Cold migration of MVD takes 2 min
- Security
    - Data storage protection: yes
    - Data transport protection: yes, but simple
    - System administration protection: yes
- Maintainability

- Analysability: All user's connection history and migration information are logged.
- Modifiability: Partially Accepted
- Testability: The test of Integrated function is OK, unit tests limited

- Overall
  - Ease of use: depending on the component to be used and on the use case itself. Some domain knowledge is required.
  - Usefulness: Very useful. All requirements have been fulfilled. Setting up the same system under the same service level objectives without BASMATI would have caused approximately 3-4 time the costs.
  - Documentation: Available.

5.2 Large Events Use Case

5.2.1 Comparing the DAS FEST demonstrator with and without BASMATI

A comparison between the DAS FEST application with and without BASMATI can only be estimated since the application was developed in the frame of the project in 2018. The application was never deployed and executed without the help of the BASMATI platform.

The main aim of the last DAS FEST prototype was a real-time visualization of the amount and movement of the festival visitors for security purposes.

The necessary adaptations of the BASMATI modules were straight forward and the deployment of the BASMATI framework itself was unproblematic and efficient. For the deployment of the DAS FEST demonstrator, all components have been described in BEAM. The required hardware- and software-resources for all components were available within 5-10 minutes. Through all 3 days of the festival, the system was operational and performant.

Criteria	DAS FEST without BASMATI (estimation)	DAS FEST with BASMATI
Ressource Utilization Small VM = 10 GB RAM; 1,4 GHz CPU Medium VM = 20 GB RAM; 1,4 GHz CPU Big VM = 40 GB RAM, 2,8 GHz CPU	Without the automatic scaling provided by BASMATI, the DAS FEST application would need the maximum amount of VMs that were needed for a real time processing of the signals which means: 3 small and 18 medium VMs for all 3 days of the festival + for testing and simulation purposes beforehand. The main problem here would have been to estimate the	The Basmati platform itself was deployed on-site on a bare metal machine having 8GB RAM, 4 Cores, 1TB disk (not used completely), in the cloud on a VM with 4GB, 2 cores, 40GB disk. The application components were scaled horizontally resulting in a minimum set of 6 VMs: 3 small VMs and 3 medium VMs. The maximum number of

	needed resources since there were to historical data about the number of WIFI signals detected. 27 Raspberry PIs having 1G RAM, 4 CPU cores and 8GB disk.	provisioned VMs would be: 3 small and 18 medium.[2] The amount of 27 Raspberry PIs and their hardware resources was the same as described in the left column.
Personal Costs for the deployment and execution of the application components	The personal costs for the deployment can only be estimated. The application requirements need to be specified as well (but not in the BEAM format) which means 1 day for all application components. Then each application component needs to be deployed on a dedicated VM which might take 1 day per component, resulting in 5 days for the minimum set. The trilaterator component would need to be deployed approx. 15 times, by copying the VM image which results in additional 1 day + 1 day to check if all components are wired together as expected. In sum ~ 7 days for the plain deployment + weeks in order to wire all components together for the proper data flow.[3]	Defining the application description in the BEAM format was a half-day task for each component for a developer knowing the BEAM meta model. The deployment was fully automatic. In sum ~ 2.5 days.
Hardware Costs	Sum of cost for the	In sum the costs for the cloud

[2] As the fixation of the on-site satellite was not stable, the data transmission during the festival was lower than expected, resulting in a non-representative scaling of the trilaterator components simply because the data was not delivered on time. In a post simulation, the trilaterators were scaled up to 15 in order to process the data in real-time.

[3] With the help of BASMATI, single components only need to specify their required input and output relative endpoints and BASMATI keeps track of the real IPs for the provisioned VMs and manages the availability of the data for the component´s processing in a flexible and automated way. The effort to handle that manually is hard to predict.

• Small VM = $0.047 per On Demand Hour • Medium VM = $0.073 per On Demand Hour • Big VM = $0.111 per On Demand	application components can be estimated as at least double of the costs we had when using BASMATI. Not considered here is the fact, that the needed resources for the processing and trilateration of the signals could not be estimated in advance because no reference data was available. Therefor the costs would have been higher in order to guarantee the processing in real time or the processing would not have been done in real time.	resources created by the DAS FEST demonstrator in July were 213,66$. With the help of BAMSATI, the cost would scale based on the number of Raspberry PIs placed on the festival area. The more detectors would be placed, the higher the number of signals to process during peak times and only during peak times.
Mean time to deploy	The provisioning takes the same time as described in the right column, as well as the deployment itself. But additional manual work would be required as described for criteria “Personal costs”.	The provisioning of the required VM takes ~ 5 min The deployment takes ~ 6 min for the most complex component of this use case and ~ 3 min for the simplest one.
QoS – real time processing	We can only give rough estimates for the QoS of the application without BASMATI. The probability is high that without BASMATI we would had problems e.g. because an adaptation of the WIFI signal detection interval was needed which without BASMATI would have been an error prone task to perform on all 27 Raspberry PIs. In addition, we expect bottlenecks in the processing of the data to occur without BASMATI.	Thanks to the scalability managed by BASMATI, the signals were processed in real time. No delays or bottleneck have been experienced in the processing and visualization of the data.
Governance	Installing, updating and monitoring the Raspberry PIs would have been a lot of manual work without	Installing, updating and monitoring the Raspberry PIs was easy due to the central BASMATI edge management.

	BASMATI. There would have been the need to login onto each raspberry pi in order to install the software which would have been needed also in case of adaptation of the configuration (e.g. adaptation of the signal detection interval). Also installing, updating and monitoring the application components would have been a lot of manual and error prone work without BASMATI. During the 3 days of the festival two-three persons would be needed for monitoring the system and fixing configurations.	Easy to manage application components thanks to one central cloud federation management interface and thanks to central monitoring of all components. During the 3 days of the festival one person was monitoring the overall system.
Scalability	Either no scalability would have been used or depending on the cloud provider, a dedicated scalability would have been introduced. Therefore, the same application could not easily be deployed and executed on different cloud providers in the future.	In BASMATI, the scalability requirements are defined in BEAM and BASMATI enacts the scaling on the cloud provider used. This means that regardless of where (Amazon, Azure, Google Cloud, OpenStack) the application is deployed, there is no need to adapt the scalability definitions.

5.2.2. Interview summary

After the successful operation of the system, CAS conducted several evaluation interviews, whose transcripts are provided in the appendix, chapter 8.3. The following section gives a summary of the interview results along the evaluation criteria defined in chapter 3.2.

- Functional suitability
    - Functional completeness: All necessary functionality was available. Even more that what was used for the purpose of the use case.

- Functional correctness: BASMATI behaved as required.

- Performance
  - Time behaviour: Transferring and computing over 7 million single data sets (~1GB of data) in these 3 days was not an issue for BASMATI and the use case components. The only bottleneck was the satellite connection.
  - Resource utilization: on-site: 8GB RAM, 4 Cores, 1TB disk (not used completely), in the cloud: 4GB, 2 cores, 40GB disk – as usual and as expected.
- Compatibility
  - Interoperability: No restrictions
  - Modularity: Easy to select, add, replace single components of BASMATI.
  - Adaptability: Manageable. Open Source Code. No dedicated extension points available.
- Usability
  - Recognizability: Good
  - Learnability: Good
  - User error protection: No
  - User interface design: Ok for the purpose, could be improved.
- Reliability
  - Availability: 100% during the festival
  - Fault tolerance: good
  - Recoverability: very good, automatic, takes 5 min
- Security
  - Data storage protection: yes
  - Data transport protection: yes
  - System administration protection: yes
- Maintainability
  - Analysability: all requests and responses are logged
  - Modifiability: good, open source, standard interface design
  - Testability: depending on the component, overall good
- Overall
  - Ease of use: depending on the component to be used and on the use case itself. Some domain knowledge is required.
  - Usefulness: Very useful. All requirements have been fulfilled. Setting up the same system under the same service level objectives without BASMATI would have been unmanageable for CAS/YM at that time.
  - Documentation: Available but distributed over many sources.

Overall the use case development, deployment and operation with BASMATI was compared to other solutions very efficient and reliable. Support from BASMATI partners was necessary for

the first deployment but could be done by any dev-op in the future. Some graphical interfaces could be improved but all relevant functionality was available.

5.3 TripBuilder Use Case

The Tripbuilder application for BASMATI consists of a single online server that contains: (i) a web front-end, (ii) a back-end knowledge base, and (iii) a software module that executes the rather complex algorithm for finding the best travelling path, and that is called for each submitted request. The query computational latency is significant and critical, as it is part of the user perceived delay and thus directly impacts the quality of experience.

Indeed, the main quality of experience (QoE) term for Tripbuilder is the Response Time (RT). The RT is the sum of two intervals: (i) the Round-Trip Time between the user issuing the request and the TripBuilder server, and (ii) the time the server takes to compute the reply. The lower the response time, the higher the interactivity perceived by the users accessing TripBuilder. The QoE level then depends on *how many* and *where* the different servers of Tripbuilder are instantiated. Naturally, there is a strict relationship between the cost (budget) and QoE. One of the main challenges is in fact to find a proper placement of the Tripbuilder server such that the QoE is maximized while respecting the budget.

5.3.1 Comparing Trip Builder with and without BASMATI

From a functional standpoint, the BASMATI platform correctly runs a TripBuilder instance and all the functionalities work as intended (Figure 9 shows an example of a TripBuilder instance running with BASMATI). During this step, minor issues regarding the encapsulation of the application description in a BEAM document need to be handled.

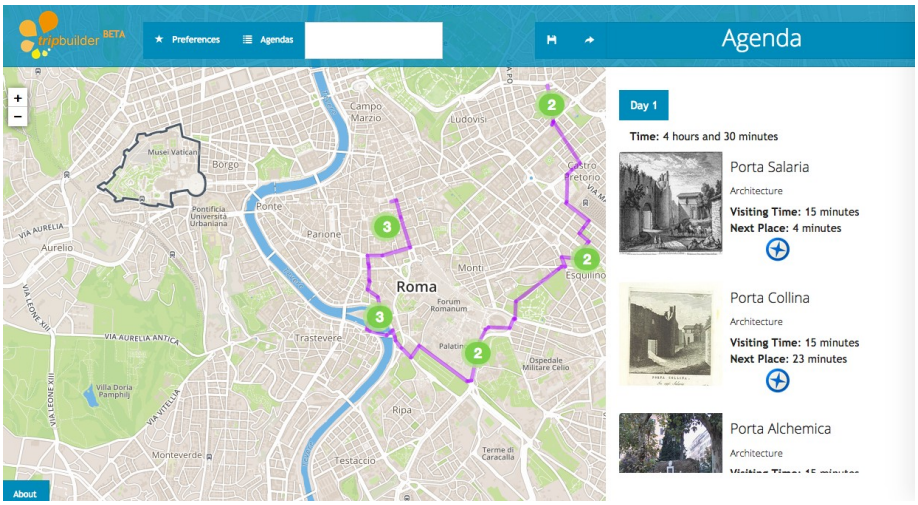

Figure 9. TripBuilder in action

Criteria	*Tripbuilder without BASMATI*	*Tripbuilder with BASMATI*
Resource Allocation	Without BASMATI, the resource utilization of Tripbuilder is substantially static, or manually performed. Basically, when an instance is under heavy load, an operator can decide to start another instance manually.	BASMATI added dynamicity to the utilization of resources. Scale up and down is done automatically according to the load of the machine, which is measured by the BASMATI monitoring platform. Therefore, an operator was no longer necessary to manage the starting of instances.
Cost	Staying inside the budget in plain Tripbuilder is managed manually. This means that an operator starts and shuts down a server to stay into the cost. This usually results in suboptimal solutions, for example due to delay in reacting to complex situations.	BASMATI allows for a more precise control of the cost with respect of the budget, thanks to the fast reaction for over/under loaded VMs, which is made possible by the integrated monitoring system.
Response Time	Without BASMATI optimizing the response time (keeping the cost fixed) is quite hard, as it requires an operator to resolve a complex optimization problem in a few seconds.	BASMATI provides algorithms with the ability of computing the allocation of the application (given a cost) that computes a near optimal allocation of the servers.
Governance on different clouds	To run an instance of Tripbuilder on top of a virtualized resource is a trivial task. However, running Tripbuilder on different cloud service providers at the same time requires to: (i) write and maintain different formats for the application; (ii) deal with many different management interfaces.	Running Tripbuilder in BASMATI requires an initial effort to port the application topology in BEAM/TOSCA. Running it on multiple cloud service providers requires no special action, resulting in a large saving of time in terms of governance.
Data Confidentiality	Data confidentiality in Tripbuilder refers to where the requests and the personal preferences of the users are stored. Enforcing to keep the user´s data in specified countries without BASMATI can be done manually, which requires specific personnel and can delay the operation of scale up.	With BASMATI the retaining of personal data of the users is done automatically following the definition specified in BEAM. This means that if one wishes to keep the data for the users in a given geographical area, it is possible to specify a set of cloud service providers in BEAM. When the support scales up, only the specified providers will be considered.

5.3.2 Automatic Scalability

To quantitatively assess the quality of experience in TripBuilder, we measured the service time of the TripBuilder server under different request´s periods. Figure 10 shows how the service time behaves when varying the period (distance between the requests) parameter. The values are computed on 15 independent runs.

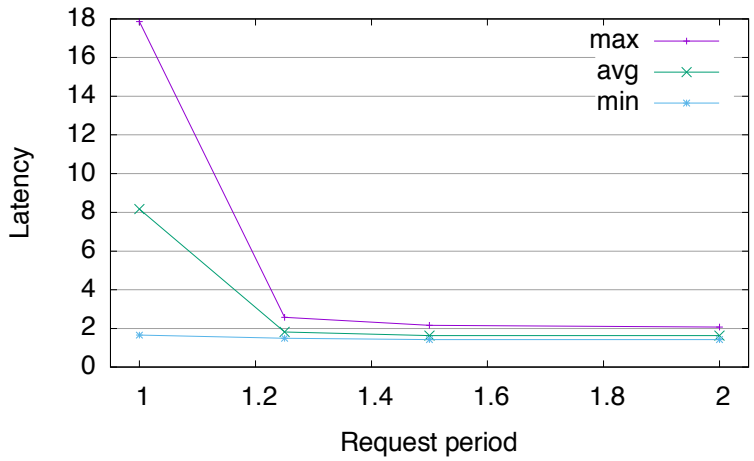

Figure 10. Latency of requests to TripBuilder. All values are in seconds.

From the image it is clear that a distance of 1.2 seconds between two consecutive requests on a TripBuilder server results in a high response time (on average > 2 seconds).

Building on the above steps, we evaluated the ability of BASMATI to deploy on multiple cloud service providers and perform load balancing. This scenario considers a situation in which two instances of TripBuilder were deployed on two different cloud service providers but only one active. Then, a request every second is submitted to the active server.

We notice that the BASMATI monitoring correctly registers the spike in the response time, and communicates it to the SLA Manger, which in turn raises a violation. After a number of violations (in this case the threshold was set to 5), the SLA Manager informs ACE which automatically adds a new TripBuilder server as well as a load balancer in front of the now two active servers. From our measurement, this indeed dropped the response time to an acceptable level of interactivity (< 2 seconds), while maintaining the same rate of requests (one every second).

5.3.2. Summary of interview results

The following section provides a summary of the interview results along the evaluation criteria.

- Functional suitability
  - Functional completeness: All designed functionalities were available.
  - Functional correctness: BASMATI behaved as expected.
- Performance
  - Time behavior: Few minutes to start an application.
  - Resource utilization: on-site: 8GB RAM, 4 Cores, 1TB disk.
- Compatibility
  - Interoperability: Supports all major OSs.
  - Modularity: Yes, BASMATI is service-based platform.
  - Adaptability: Module are open-source and they can be modified.
- Usability
  - Recognizability: Good
  - Learnability: Good
  - User error protection: No.
  - User interface design: Many interfaces, some good other to be improved.
- Reliability
  - Availability: satisfactory.
  - Fault tolerance: good
  - Recoverability: fast in regular occasions, longer (week) when serious flaws arise
- Security
  - Data storage protection: yes
  - Data transport protection: yes
  - System administration protection: yes
- Maintainability
  - Analysability: general and per-component log
  - Modifiability: good, open source, standard interface design
  - Testability: component dependent
- Overall
  - Ease of use: component dependent, some domain knowledge is required.
  - Usefulness: Yes
  - Documentation: Available

6 Component level evaluation

6.1 Application Controller Evaluation

6.1.1 Added value

- AC deploys and manage the applications according to the Deployment Documents which has optimal plans for the applications' needs and current available resources.
- AC is responsible for processing complex procedures of application's lifecycle management while providing a simple REST-based interface.
- AC provides scaling up and down interfaces for deployed applications too.

6.1.2 Technical Aspects Evaluation

- AC can deploy applications on federated clouds environment less than 200 secs
- AC can scale up applications less than 150 secs

6.1.3 Project Objectives

- **Objective - 5**: Ultra-scalable hybrid infrastructure management for mobile heterogeneous resources and cloud federations
    - AC can deploy and manage applications running on federated cloud environment.
    - AC supports scaling up and down of application's resources on runtime.

6.2 Resource Broker Evaluation

6.2.1 Added value

- RB generates lists of best-fit candidates among available federated cloud resources.
- RB considers multiple requirement facets like price, required specification, region, availability, and previous performance. RB sorts the list based on the price.

6.2.2 Technical aspects evaluation

- RB can generate a list of best-fit candidates meet requirements in less than 10 secs
- RB can accept multiple requirement conditions like price, hardware and software specification, and region.

- In addition to sorting the candidates based on price, RB is giving performance rating to each resource to aid Decision Maker in making final decision on which resources to be utilized. (not really implemented yet)

6.2.3 Business aspects evaluation

- RB sorts the order of candidates based on the price, so it can optimize the cost while not degrading QoS of the application.

6.2.4 Project Objectives

- **Objective - 4**: Dynamic brokerage for placement and offloading through runtime adaptable deployment patterns of the brokerage platform
    - RB provides list of best-fit resources considering locations, price, and performance requirements, therefore, DM can make optimized deployment plans with helping of RB.

6.3 Federation Monitoring Evaluation

6.3.1 Added value

- FM is not dependent on CMP (Cloud Management Platform) and can monitor the VM resources deployed in CMP managed cloud.
- The monitoring data collector used in FM is dynamically scalable regardless of the number of federated clouds.
- All monitoring data collected by FM is provided in API format and provided by CSP (Cloud Service Provider) / VM / time / monitoring metric item type.

6.3.2 Technical aspects evaluation

- FM collects 28 monitoring items of VM.
- FM collects monitoring data every 15 seconds.
- FM provides real-time monitoring (less than 3 seconds) for immediate change detection (not used on the BASMATI platform).

6.3.3 Business aspects evaluation

- Improve operational management efficiency by leveraging production to optimize resource utilization in multi-cloud infrastructure environments with multiple clouds

6.3.4 Project Objectives

- **Objective - 5**: Ultra-scalable hybrid infrastructure management for mobile heterogeneous resources and cloud federations
    - FM collects the same level of monitoring items in federated cloud environment and provides monitoring in each cloud environment / VM unit
    - FM collects VM resource usage as monitoring data and is provided as an API. AC can scale up / down applications based on monitoring data.

6.4 Decision Maker Evaluation

6.4.1 Added value

The Decision Maker (DM) module:

- Generates plans for the application geo-placement
- It coordinates other components (such as the Knowledge Extractor and the Resource Broker) to retrieve information as account requirements, previsions as well as all the available information on the footprint generated by applications on resources during their past executions.
- Takes major adaptive offloading decisions at runtime that were not resolved at provider level.

6.4.2 Technical aspects evaluation

- The DM can perform placement decision considering > 20 different cloud service providers
- A placement plan of an application is generated in < 60 seconds.

6.4.3 Project Objectives

- **Objective - 1**: Mobile cloud services enablement and portability to exploit brokerage and offloading to cloud federations and / or other devices (i.e. device-to-device) based on application typology

- DM supports the generation of placement of applications considering different computational resources, by considering their ability in supporting the application. With a proper resource modeling, DM can generate placement plans that span from different flavors of cloud resources to mobile devices.

- **Objective - 4**: Dynamic brokerage for placement and offloading through runtime adaptable deployment patterns of the brokerage platform
  - The DM coordinates the gathering of context information regarding applications from other BASMATI components, such as Knowledge Extractor and Resource Broker. This information is necessary to provide runtime placement and offloading solutions.
  - DM is able to provide different application placement plans when fed with further information about the applications. For example, if a selected provider is underperforming and this information arrives at the DM, the DM will provide a different plan discarding the provider.

6.5 Data Management Framework Evaluation

6.5.1 Added value

The BASMATI Unified Data Management Framework (BUDaMaF) is managing the data and data stores used in the context of the BASMATI federation and the hosted applications, at various degrees as needed by each use case.

In detail it provides the following services as Rest APIs:

- Data store scalability using the BASMATI federation.
- Data migration from one physical location to another.
- Data replication.

Moreover it provides an obscuration layer, hiding to a large degree the actual data servers and data stores as well as their technologies, so that users know only what they need to know and work more efficiently.

Finally, it adds a security and anonymization layer to the data traveling through it, enforcing the pre-configured policies according to the data type, the data owners and the physical location of the data.

6.5.2 Technical aspects evaluation

Aspect	KPI	Target	Description
Real time scaling	Scaling Response Time	<= 3 minutes [4]	The real time scaling is used to achieve maximum resource usage efficiency while handling the usage curves and usage spikes for a hosted application.
Resource Management	Average % of resource usage per 10 minutes	>90% [5]	This aspect covers the maximization of resource usage effectiveness as well as the minimization of the costs for unused resource binding.
SLA response time violations minimization	% of request out of specified time	<1%	This aspect covers the minimization of SLA violation costs for the hosted application, showing the effectiveness of the real time scaling and data migration services.

6.5.3 Project Objectives

- **Objective - 1**: Mobile cloud services enablement and portability to exploit brokerage and offloading to cloud federations and / or other devices (i.e. device-to-device) based on application typology
    - BUDaMaF enables the migration as well as the transformation of data from one data store to another, following the needs of mobile cloud services that depend on these data. Also, it can be used to create data stores in edge devices, incorporating them to the mixed cloud schema, enhancing the portability and resource usage.
- **Objective - 5**: Ultra-scalable hybrid infrastructure management for mobile heterogeneous resources and cloud federations

[4] In[CITATION AlQ16 \l 1031] [1] a 4 minute VM provision time has been achieved, we aim to achieve scaling time of 3 minutes or less, countering a plethora of usage spike problems.
[5] "… in over 90% of cases, users tend to overestimate the amount of resources that they require, wasting in some cases near to 98% of the requested resource." [CITATION Mor13 \l 1031][2]

- BUDaMaF can scale, destroy and create data store clusters in semi-real time using machines connected to the machine hosting its Core service. This means that it can create data stores using any infrastructure available as long as it can access it using the SSH protocol, enabling it to manage complex data store architecture dynamically using hybrid architectures which involve cloud, edge and mobile resources.

- **Objective - 6**: Multi-provider and multi-tenancy functional and non-functional guarantees for mobile cloud services
  - Through its security and anonymization module, BUDaMaF can provide security and privacy guarantees for all data traveling through it. This, though, does not include any data coming in the framework, which should be covered by the data owner. The data coming out of the framework can be encrypted or anonymized according to the needs of each use case.

6.6 Knowledge Extractor

6.6.1 Added value

The Knowledge Extractor (KE) component provides useful insight to the Decision Maker in order to generate plans and take offloading decisions in an efficient and efficacy way. The KE provides insight concerning the user mobility, the application monitoring and the knowledge that has been acquired from previous decision plans. The added value resulting from use of the KE comprises:

- Prediction of the number of users that are gathered at Points of Interest.
- Prediction of the resource demands of the cloud deployed applications
- Evaluation of the candidate deployment plans.

6.6.2 Technical Aspects Evaluation

KE functionalities have been implemented using machine learning techniques and covers the Basmati project objectives 1, 2 and 4.

KE is designed and developed following a generic and inherent adaptively mechanism that support the portability of any type and number of service stereotypes. This fact satisfies the Key Performance Indicator (KPI) of objective 1.
The KPI of objective 2 is also satisfied. The situational/context awareness mechanism latency is less than 10 seconds. The response of KE is real time.
KE does not use one specific prediction model but a meta-model that involves a set of prediction techniques and provide outcomes concerning application placement and offloading. These prediction models that constitute the KE meta-model are more than five and more models can be added in a plug and play fashion. This design of KE satisfies the KPI of objective 4.
In general, KE involves supervised machine learning techniques that use knowledge acquired from previous observations and make predictions in upcoming instances. We made experiments in different use cases and the accuracy of the prediction outcomes goes beyond the 87%. This verifies the applicability and adequacy of the KE component.

6.6.3 Project Objectives

- **Objective - 1**: Mobile cloud services enablement and portability to exploit brokerage and offloading to cloud federations and / or other devices (i.e. device-to-device) based on application typology
    - KE provides functionalities that aim at the enablement of mobile cloud services. KE implements a set of predictive techniques for resource demands as a function of the current status of the applications, the location of users that use the applications and time properties. The prediction outcomes are based on the analysis and the models that identify the patterns of application usage. The outcomes will be used from the Decision Maker to adapt the applications in order to satisfy the portability and mobility perspectives.

- **Objective - 2**: Situational knowledge acquisition and mobility patterns understanding providing the ground for real-time adaptations of federation and brokerage decisions
    - The behaviour of users and group of users affect the provision of mobile cloud services. KE records the impact of user behaviours and applications nature into the resource usage. These records constitute a knowledge base with which we can evaluate any candidate deployment plan. KE can detect the most similar deployment plans that are recorded in the knowledge base and using a similarity technique to evaluate the applicability of the current deployment plan.
- **Objective - 4**: Dynamic brokerage for placement and offloading through runtime adaptable deployment patterns of the brokerage platform
    - The Decision Maker provides plans concerning the runtime placements and offloading solutions using context information from the KE and other Basmati components. KE can estimate the resources that an application needs function time periods in order the brokerage take place in a dynamic and elastic way.

6.7 Cloud Federation Controller

6.7.1 Added value

This component allows fully automated resource sharing between discrete commercial instances of the BASMATI platform.

In the case of the DAS FEST Use Case the edge resources offered by the onsite Das Fest Edge Controller BASMATI platform were reserved and deployed through the federation management controller by the in-cloud BASMATI Application controller platform.

The same technology could be employed for the construction of a federation of cooperating cloud service providers for deployment of geo-placement cloud resources in support of the Mobile Video Desktop Use case.

6.7.2 Technical aspects evaluation

The cloud federation controller presents an industry standard OCCI REST API for the declaration, configuration and activation of cloud federation relationships between BASMATI platforms.

6.7.3 Business aspects evaluation

The description of the offer of service and the associated conditions of a federation relationship is described using service level agreements based on the industry standard WS Agreement.

6.7.4 Project Objectives

- **Objective - 2**: Situational knowledge acquisition and mobility patterns understanding providing the ground for real-time adaptations of federation and brokerage decisions
  - RB provides list of best-fit resources adapting to the real-time changes/downtime/improvement in resources in federation.
- **Objective - 3**: Development, deployment and configuration of abstracted and integrated federated cloud environments based on multi-objective optimized federation logic
  - This component delivers the complete cloud federation management component described by this objective allowing the construction and management of a variety of fully automated cloud federation configurations as described in the deliverable D4.1.
- **Objective - 4**: Dynamic brokerage for placement and offloading through runtime adaptable deployment patterns of the brokerage platform
  - The component cooperates with the resource management components providing information describing provider resource quota in support of both commercial and technical brokerage decisions and actions.

6.8 BEAM Document Processor

6.8.1 Added value

This component and the associated BEAM / TOSCA technical reference specification allows transformation of an application description to the corresponding WS Agreement document for service life cycle control and its associated technical configuration manifest and deployment scripts.

In the case of the DAS FEST Use Case the application description was prepared as a BEAM document suite and processed by the BEAM document processor during application deployment allowing subsequent control over the application life cycle.

The BEAM document processor was also used for the description, deployment and control of the application components required for the Mobile Video Desktop Use case.

The BEAM document processor was also used for the description, deployment and control of the application components required for the Trip Builder Use case.

6.8.2 Technical aspects evaluation

The BEAM document processor and its associated BEAM / TOSCA specification is an XML format based on and extending the international standard TOSCA. New service template TAG types have been added for the description and control of provider selection and resource placement along with extensions allowing service level guarantees to be described and associated with deployed resources.

6.8.3 Business aspects evaluation

The BEAM / TOSCA processor produces a customer service level agreement, in WS Agreement format, through which the business aspects of the operation are formulated.

6.8.4 Other aspects evaluation

This could be a suitable candidate for a standardization initiative.

6.8.5 Project Objectives

- **Objective - 1**: Mobile cloud services enablement and portability to exploit brokerage and offloading to cloud federations and / or other devices (i.e. device-to-device) based on application typology
    - The BEAM document format allows aggregation of information relating to resource placement during the initial phases of the BASAMTIZE process.
- **Objective - 2**: Situational knowledge acquisition and mobility patterns understanding providing the ground for real-time adaptations of federation and brokerage decisions
    - RB provides list of best-fit resources adapting to the real-time changes/downtime/improvement in resources in federation.
- **Objective - 3**: Development, deployment and configuration of abstracted and integrated federated cloud environments based on multi-objective optimized federation logic
    - This component transforms the output of the BASMATIZE process for the deployment and management of the subsequent cloud application.
- **Objective - 4**: Dynamic brokerage for placement and offloading through runtime adaptable deployment patterns of the brokerage platform
    - The BEAM document format is an ideal pivot around which both commercial and technical brokerage activities may be coordinated.
- **Objective - 5**: Ultra-scalable hybrid infrastructure management for mobile heterogeneous resources and cloud federations
    - The BEAM document format and processor allow definition of scalability requirements, limits, thresholds and the conditions in which they will be engaged.

6.9 EDGE Resource Management

6.9.1 Added value

This component performs fully automated attachment of EDGE device resources for use by cloud application instances that require a geo-localised onsite hardware presence for data collection and distribution.

In the case of the DAS FEST Use Case the edge resources offered by the onsite Das Fest Edge Controller version of the BASMATI platform were used for the collection of crowd mobility data through a network of Raspberry Pi devices provisioned and attached through the Edge controller in response to requests received by the federation management controller issued from the in-cloud BASMATI Application controller platform.

6.9.2 Technical aspects evaluation

The edge resource management component presents a collection of industry standard OCCI REST APIs that allow integration with the BASMATI platform and provide for configuration of the device collections and management of their allocation.

6.9.3 Business aspects evaluation

The edge resource management component would facilitate IOT device allocation and management for use in a variety of commercial applications where onsite presence is required.

6.9.4 Project Objectives

- **Objective - 1**: Mobile cloud services enablement and portability to exploit brokerage and offloading to cloud federations and / or other devices (i.e. device-to-device) based on application typology
    - This component delivers the means to perform offloading of cloud application processing and data collection allowing this to be performed by allocated edge devices and subsequently shortening the costly last mile in the cloud computing paradigm.
- **Objective - 2**: Situational knowledge acquisition and mobility patterns understanding providing the ground for real-time adaptations of federation and brokerage decisions
    - RB provides list of best-fit resources adapting to the real-time changes/downtime/improvement in resources in federation.
- **Objective - 4**: Dynamic brokerage for placement and offloading through runtime adaptable deployment patterns of the brokerage platform
    - The component cooperates with the resource management components providing information describing provider resource quota in support of both commercial and technical brokerage decisions and actions.

6.10 Cloud Provider Management

6.10.1 Added value

This component allows fully automated deployment and coordination of cloud resources for use in the construction and management of complex business application scenarios. The component provides a generic abstraction of the cloud resource usage allowing deployment through any of the current public or private cloud management interfaces. This allows deployment of the application resources to be performed in precise geographical regions as covered by the combined collection of availability zones offered by the different public and private cloud providers.

In the case of the DAS FEST Use Case this component is responsible for the control and management of the complete application lifecycle including the deployment and management of the cloud resources required for the signal ordering, load balancing, trilateration and storage and aggregation components onto the amazon EC2 cloud.

The same technology is also employed for the geo-placement of the virtual desktop resources in both south Korean private data centres and European public data centres under control of the decision processes of the Mobile Video Desktop Use case.

6.10.2 Technical aspects evaluation

The cloud provider management component presents a complete collection of industry standard OCCI REST APIs allowing declaration, configuration and activation of cloud resource discovery, selection, placement, deployment and management.

6.10.3 Business aspects evaluation

This component exposes business-oriented information, including pricing and policy, through provider level service agreements. This information is exploited by the resource selection and brokerage components providing economic guidance to the process through which the application description is prepared.

6.10.4 Project Objectives

- **Objective - 1**: Mobile cloud services enablement and portability to exploit brokerage and offloading to cloud federations and / or other devices (i.e. device-to-device) based on application typology
    - This component is at the operational heart of the brokerage and offloading mechanisms in response to requests for placement and deployment issued by the application controller.
- **Objective - 2**: Situational knowledge acquisition and mobility patterns understanding providing the ground for real-time adaptations of federation and brokerage decisions

- RB provides list of best-fit resources adapting to the real-time changes/downtime/improvement in resources in federation.

- **Objective - 3**: Development, deployment and configuration of abstracted and integrated federated cloud environments based on multi-objective optimized federation logic
  - This component coordinates with the cloud federation management component for the declaration, discovery and deployment of services and resources within the resulting cloud federation.
- **Objective - 4**: Dynamic brokerage for placement and offloading through runtime adaptable deployment patterns of the brokerage platform
  - The component coordinates with the resource management components providing information describing provider resource quota in support of both commercial and technical brokerage decisions and actions.
- **Objective - 5**: Ultra-scalable hybrid infrastructure management for mobile heterogeneous resources and cloud federations
  - This component provides the basis of the provider management system in accordance with the standard OCCI oriented architecture allowing dynamic replication of the service components as required to achieve the desired degree of operational scalability.
- **Objective - 6**: Multi-provider and multi-tenancy functional and non-functional guarantees for mobile cloud services
  - This component encapsulates the complete collection of cloud provider interfaces allowing multi–cloud deployment respecting application and geo-placement requirements.
- **Objective - 7**: Effectiveness of BASMATI outcomes validated and communicated through three real-world scenarios from different application domains
  - This component is involved in all three use cases for the management of the application life cycle and the deployment of all cloud and edge resources.

6.11 Decision Maker Evaluation

6.11.1 Added value

The Decision Maker (DM) module:

- Generates plans for the geo-placement of application components.
- coordinates with other components (such as the Knowledge Extractor and the Resource Broker) to retrieve information such as account requirements, previsions as well as all the available information on the footprint generated by applications on resources during their previous executions.
- Takes major adaptive offloading decisions at runtime that were not resolved at provider level.

- Uses an AI method (Genetic Algorithm) to deal with the extremely large solution space found in cloud federations and diverse applications.

6.11.2 Technical aspects evaluation

- The DM can perform placement decisions considering > 20 different cloud service providers and 5 different optimization parameters
- A placement plan of an application is generated in < 60 seconds.
- The characteristics of the algorithm is that it is stable (i.e., it converges to the optimal solution), robust, effective, and achieves a Pareto-optimal solution through scalarization of multi-objectives.

6.11.3 Business aspects evaluation

- DM combines 5 different optimization criteria, including cost, distance to the user, and distance between application nodes, helping cloud providers to address the needs of their customers.

6.11.4 Project Objectives

- **Objective - 1**: Mobile cloud services enablement and portability to exploit brokerage and offloading to cloud federations and / or other devices (i.e. device-to-device) based on application typology
    - DM supports the generation of placement of applications onto different computational resources, by considering their ability in supporting the application. With a proper resource modeling, DM can generate placement plans that span from different flavors of cloud resources to mobile devices.
- **Objective - 4**: Dynamic brokerage for placement and offloading through runtime adaptable deployment patterns of the brokerage platform
    - The DM coordinates the gathering of context information regarding applications from other BASMATI components, such as Knowledge Extractor and Resource Broker.
    - DM is able to provide different application placement plans when fed with further information about the applications. For example, if a selected provider is underperforming and this information arrives at the DM, the DM will provide a different plan discarding the provider.
    - DM uses application footprint data and the inter-node relationship between applications for providing runtime placements.
- **Objective - 7**: Effectiveness of BASMATI outcomes validated and communicated through three real-world scenarios from different application domains
    - DM is used in the MVD use case.

6.12 Federation SLA Manager

6.12.1 Added value

The SLA management module plays an important role within the BASMATI architecture, being the mechanism where through which a user may describe and enforce the guarantees around their Business Level Objectives (BLOs).

The **Basmati SLA Manager** will receive the documents describing the **Customer's requirements** and will perform their preliminary processing and their storage in the **Application Repository.** Service functional requirements are captured by the Federation Controller module who may work in combination with the SLAM.

The Service Level Agreement Manager (SLAM) components within BASMATI architecture are responsible for the management activities associated with the binding contracts between the Application Controller module and both federated and non-federated Cloud Service Providers (CSPs) managed by the AMENENSIK cloud module. The set of activities, which allow the establishment of an agreement between consumers and providers, includes the definition of SLA templates, the agreement negotiation and the agreement evaluation. This final activity will consider monitoring information provided by the Monitoring Provider module within BASMATI as well as monitoring information coming from monitoring probes and provider information at application level. After evaluation, recovery actions can be triggered in case of violation detection.

6.12.2 Technical Aspects Evaluation

- The SLAM is able to provide custom contract templates based on the international standard WS-Agreement.
    - As part of the templates, customer requirements from the BASMATI platform can be specified.
- The SLAM provides the mechanism to perform the negotiation between the entities involved in the agreement.
- Work in conjunction with the BASMATI Service Description topology based on TOSCA specs.
    - Support federated scenarios thanks to the specialization of the standard specifications used.
- The component is able to receive incident signaling through the published OCCI endpoint available for other BASMATI components.
    - It is able to inject incident signalling (instead of monitoring information) from both the infrastructure and the application layers.

- It is able to trigger penalties associated to the incidents captured at runtime.

6.12.3 Business aspects evaluation

The description of the offer of service and the associated conditions of a federation relationship is described using service level agreements based on the industry standard WS Agreement.

6.12.4 Project Objectives

- **Objective - 1**: Mobile cloud services enablement and portability to exploit brokerage and offloading to cloud federations and / or other devices (i.e. device-to-device) based on application typology
- **Objective - 2**: Situational knowledge acquisition and mobility patterns understanding providing the ground for real-time adaptations of federation and brokerage decisions
  - RB provides list of best-fit resources adapting to the real-time changes/downtime/improvement in resources in federation.
- **Objective - 3**: Development, deployment and configuration of abstracted and integrated federated cloud environments based on multi-objective optimized federation logic
  - This component allows to define the contract between the different types of entities involved in the clod federation agreements. This component can work in combination with other BASMATI modules to overcome this objective.
- **Objective - 4**: Dynamic brokerage for placement and offloading through runtime adaptable deployment patterns of the brokerage platform
  - This component cooperates with the resource management policies by providing runtime information associated to the incident/violations occurred during the execution of the service. Based on the type of incident and the number of occurrences the SLA modules forward the associated penalties to the management modules in order to trigger the appropriate corrective actions.
- **Objective - 8**: Exploitation, effectiveness and standardisation of BASMATI outcomes through tangible identified plans for a wide set of BASMATI technologies-
  - Exploitation: The SLA framework adapted in BASMATI is included in Atos Research and Innovation (ARI) your CIM (Cloud Infrastructure Management) Research Incubator.
  - Standarisation: The module supports the international standard known as WS-Agreement. The contracts and agreements generated by the component become part of the BEAM (application model) used in BASMATI.

7 Conclusions

This deliverable presents the quality evaluation results of the components of the BASMATI platform. Besides the evaluation methodologies and the evaluation execution plans for each use cases, the deliverable provides the component level assessment results. The questionnaires and interviews of BASMATI evaluation results are included in appendix.

The component level assessment shows how each component played a role in achieving the objectives of the BASMATI project and how the results were met with the goals of the project.

The evaluation the BASMATI platform as a whole was conducted using 3 use cases; MVD, DAS FEST and Tripbuilder. To evaluate deeper and not miss facts not shown through numbers, we carried out fuller interviews with those who understand the intentions and goals of the BASMATI project.

As a result, we conclude the BASMATI platform could support the requirements of all use cases and it achieved the major goals we stated in the proposal.

However, the interviews revealed that some parts have to be improved before BASMATI is commercialized. To use the BEAM format and the UI of BASMATI properly, users need more extensive manuals and training in order to learn about the subject beforehand. Current user interfaces of BASMATI should be improved and rendered more intuitive to meet standard commercial level requirements. Some components require enhanced security technologies to ensure the platform's integrity and stability.

8 Appendix: Interviews and Questionnaires

8.1 Questionnaire and Interview Template

Criteria	Attribute	Questions
Functinal suitabiliy	Completeness	Does the system adequately cover the needs of the application? Does it accommodate the platform requirements of the application (e.g. operating system, Java environment, etc.)?
	Answer	
	Correctness	Does the BASMATI system behave as expected? Does it behave correctly?
	Answer	
Performance	Time behavior	How long does it take to deploy a normal mid-size application (e.g. having 2 comonents running on two VMs)? Are scale-in and scale-out events handled in time so that a good time behavior of the deployed application is guaranteed?
	Answer	
	Resource utilization	How much hardware (harddisk space, RAM, CPU) are needed for the whole BASMATI framework itself to run?
	Answer	
Compatibility	Interoperability	Does BASMATI support the deployment of any application with any OS and hardware requirements? Which restrictions are already known?
	Answer	
	Modularity	Can different modules of BASMATI be used in isolation?
	Answer	
	Adaptability	Can single modules of BASMATI be adapted? Are extension points provided to do so? Is the source code well-documented or self-explanatory?
	Answer	
Usability	Recognisability	Do the component interfaces follow standard patterns?
	Answer	
	Learnability	Are the interfaces (UI and API) laid out logically, allowing the system to be learned quickly?
	Answer	
	Error Protection	Does the system prevent incorrect parameter values

		from being specified? Does the system timely identify any incompatible configurations that would prevent the correct deployment/operation of the cloud application?
	Answer	
	User interface design	Is the user interface pleasing? Is it well designed?
	Answer	
Reliability	Availability	Please report on the conducted monitoring.
	Answer	
	Fault tolerance	Did you experience downtime of the platform during your tests? If yes how many times? How many errors were raised without affecting your experience with the platform? If yes, please describe the errors. How many fatal errors were raised?
	Answer	
	Recoverability	Please report on the mean time to recover BASMATI as well as to recover your application (components).
	Answer	
Security	Data storage	Is it possible to access the application data? What kind of security mechanisms are in place in order to secure your applications data? Can your own custom security mechanisms be use to secure our application?
	Answer	
	Datatransport	Which mechanisms are in place to secure the data transport (a) between your application components and (b) between your components and BASMATI?
	Answer	
	System administration	Which mechanisms are in place to securely access (a) BASMATI provisioned VMs and (b) the BASMATI administration UI?
	Answer	
Maintainability	Analysability	Can errors and bugs easily be traced? Are sufficient logs provided?
	Answer	
	Modifiability	Is the source code available for everyone? Can bugfixes be committed by everyone? If the answer is “no”: Who is in charge of doing so?
	Answer	
	Testability	Can the whole BASMATI framework be tested or single components? Is it clear how to test the components? Please report on the test coverage if possible.
	Answer	
Overall	Ease of Use	Are the components easy to install, configure, and

		use?
	Answer	
	Usefulness	Were you able to operate your application with the BASMATI services?
	Answer	
	Documentation	Is a documentation available covering of all BASMATI components that the user needs to interact with? Do you consider the documentation useful?
	Answer	

8.2 Questionnaires and Interview Results from ETRI

8.2.1 First interview

Criteria	Attribute	Questions
Functinal suitabiliy	Completeness	Does the system adequately cover the needs of the application? Does it accommodate the platform requirements of the application (e.g. operating system, Java environment, etc.)?
	Answer	*Yes, it supported MVD Connection Manager and Host applications' requirements completely. We needed host-passthrough virtualization technology and CentOS 7 linux for the Host VMs, and BASMATI allocated right specced VMs.*
	Correctness	Does the BASMATI system behave as expected? Does it behave correctly?
	Answer	*It took some hours to configure MVD's BEAM document correctly,but we could make the deployment for our application fit for our requirements on BASMATI platform.*
Performance	Time behavior	How long does it take to deploy a normal mid-size application (e.g. having 2 comonents running on two VMs)? Are scale-in and scale-out events handled in time so that a good time behavior of the deployed application is guaranteed?
	Answer	*When we deployed 1 CM and 2 Host configuration, it took less than 5 minutes. But in installation process for the Host, it needed a 15GB golden disk image file to be downloaded. Considering the downloading time, deployment performance of BASMATI is fine for us. Same goes to scaling out case. We used scaling out only for Host. So, the golden disk image file must be downloded when scaling up Host, too. Scaling-out took similar time with deployment of Host VMs, less than 5 minutes.*
	Resource utilization	How much hardware (harddisk space, RAM, CPU) are needed for the whole BASMATI framework itself to run?
	Answer	*After the deployment of MVD, it took negligible amount of resources for BASMATI related components, like monitoring agent. It was less than 1% of CPU, 10MB of Memory, and 20MB of disk space.*
Compatibility	Interoperability	Does BASMATI support the deployment of any application with any OS and hardware requirements?

		Which restrictions are already known?
	Answer	*We used Centos 7.x and Ubuntu 14 for our usecase and there was no problem to use them. And we required very big VMs for Hosts, and we were supplied with 12-core, 16-GB VMs that was sufficient for us.*
	Modularity	Can different modules of BASMATI be used in isolation?
	Answer	*All components of BASMATI use REST API and can be executed seperately by design. But, they use dedicated REST API and depend on each other. Some modules can be used independantly if they are modified.*
	Adaptability	Can single modules of BASMATI be adapted? Are extension points provided to do so? Is the source code well-documented or self-explanatory?
	Answer	*Most modules of BASMATI has been designed to communicate with BASMATI's internal modules* *The source codes of most components have well described explanation.*
Usability	Recognisability	Do the component interfaces follow standard patterns?
	Answer	*All components support REST interface. And XML and JSON are used as exchanging data formats.*
	Learnability	Are the interfaces (UI and API) laid out logically, allowing the system to be learned quickly?
	Answer	*In user's perspective, the API and UI provide predefined procedures like one-by-one steps. However, the BEAM document format is relatively difficult to learn and understand.*
	Error Protection	Does the system prevent incorrect parameter values from being specified? Does the system timely identify any incompatible configurations that would prevent the correct deployment/operation of the cloud application?
	Answer	*When we specified wrong options in API's arguments or BEAM documents, BASMATI returns error messages and it kept running normally.*
	User interface design	Is the user interface pleasing? Is it well designed?
	Answer	*As a research project, it provides only basic interface not facy one. However, it is working and fit for BASMATI architecture.*
Reliability	Availability	Please report on the conducted monitoring.
	Answer	*We have conducted multiple times of long-run test of our use case. The longest one is almost 2 weeks and there was no critical problem.*

	Fault tolerance	Did you experience downtime of the platform during your tests? If yes how many times? How many errors were raised without affecting your experience with the platform? If yes, please describe the errors. How many fatal errors were raised?
	Answer	*Sometimes, some components got errors. Some modules restart it self within 5 minutes or less, but some modules need manual restart to recover.*
	Recoverability	Please report on the mean time to recover BASMATI as well as to recover your application (components).
	Answer	*For BASMATI components, it was less than 5 minutes average. For MVD use case, it took more than that because of the download time for big golden disk image file (>15GB)*
Security	Data storage	Is it possible to access the application data? What kind of security mechanisms are in place in order to secure your applications data? Can your own custom security mechanisms be use to secure our application?
	Answer	*We are using common security policies. Like closing unused ports and stop unnecessary services. We don't use special methods like encryption of user data.*
	Datatransport	Which mechanisms are in place to secure the data transport (a) between your application components and (b) between your components and BASMATI?
	Answer	*We are using ssh and scp with private key for accessing other components of MVD and transferring data among them.* *We provide authentication for accessing BASMATI components' REST API.*
	System administration	Which mechanisms are in place to securely access (a) BASMATI provisioned VMs and (b) the BASMATI administration UI?
	Answer	*We provide ssh RSA key for all VMs.* *BASMATI UI provides ID and password type authentication.*
Maintainability	Analysability	Can errors and bugs easily be traced? Are sufficient logs provided?
	Answer	*Actually, this needs to be improved. We needed a lot of time to create right BEAM documents for MVD because of less and ambiguous error messages.*
	Modifiability	Is the source code available for everyone? Can bugfixes be committed by everyone? If the answer is "no": Who is in charge of doing so?
	Answer	*Currently, all source codes of BASMATI modules are available for BASMATI members only.*

		After this project ends, there is possibility to open the source codes to public.
	Testability	Can the whole BASMATI framework be tested or single components? Is it clear how to test the components? Please report on the test coverage if possible.
	Answer	*We described detailed deployment and testing procedures for each module in deliverables.*
Overall	Ease of Use	Are the components easy to install, configure, and use?
	Answer	*Yes, most components are using docker container technology, so it is easy to install and run.* *To configure, it needs to reference the deployment and testing procedure documents.*
	Usefulness	Were you able to operate your application with the BASMATI services?
	Answer	*Yes, MVD application was running successfully during test period.*
	Documentation	Is a documentation of all BASMATI components the user need to interact with available? Do you consider the documentation useful?
	Answer	*There is a deliverable about deployment and testing procedure.* *I think users can have many help from that.*

8.2.2 Second interview

Criteria	Attribute	Questions
Functinal suitabiliy	Completeness	Does the system adequately cover the needs of the application? Does it accommodate the platform requirements of the application (e.g. operating system, Java environment, etc.)?
	Answer	*Host-passthrough is an indispensable feature for the Host of MVD application and we were provided VMs with that feature without problem.* *Actually that feature is not common in giant cloud providers, so all VMs provided were from Openstack-based providers.*
	Correctness	Does the BASMATI system behave as expected? Does it behave correctly?
	Answer	*Yes, we were provided the VMs installed with previously provided installation script.* *Also VMs were well placed as we specified in BEAM document.*
Performance	Time behavior	How long does it take to deploy a normal mid-size application (e.g. having 2 comonents running on two VMs)? Are scale-in and scale-out events handled in time so that a good time behavior of the deployed application is guaranteed?
	Answer	*Considering MVD application takes a lot of time for donwloading its contents from internet, the deployment time of MVD on BAMATI platform which was about 10~15 minutes were acceptable.* *Scale-out of Hosts of MVD took less than 10 minutes and Scale-in took only 1 or 2 minutes. It was predictable and sufficent for our purpose.*
	Resource utilization	How much hardware (harddisk space, RAM, CPU) are needed for the whole BASMATI framework itself to run?
	Answer	*For nomal VMs, BASMATI framework's resource footprint is small to be able to be ignored.* *However, since additional one node is mandatory to support scalability, it is a little burdensome.*
Compatibility	Interoperability	Does BASMATI support the deployment of any application with any OS and hardware requirements? Which restrictions are already known?
	Answer	*We need several very big VM instances with CentOS 7.x as OS and were provided with CentOS 7-1703 based 12-core and 16 GB-ones. It was OK for MVD use*

		case. *We also need host-passthrough feature of Host VMs, but this is not common among big public cloud providers like Amazon, Google and Microsoft. Therefore, we have been provided VMs from openstack-based providers.*
	Modularity	Can different modules of BASMATI be used in isolation?
	Answer	*All APIs of BASMATI modules are in HTTP REST format. It means each module can run independently and be called by any others that knows the specification of the API.*
	Adaptability	Can single modules of BASMATI be adapted? Are extension points provided to do so? Is the source code well-documented or self-explanatory?
	Answer	*Most modules can be used for other purpose outside of BASMATI beacuse they are supporing REST API. The source codes have well-written description.*
Usability	Recognisability	Do the component interfaces follow standard patterns?
	Answer	*Yes, they use REST API and common data formats like JSON and XML.* *Also Cloud Management Platform comprises OCCI standards.*
	Learnability	Are the interfaces (UI and API) laid out logically, allowing the system to be learned quickly?
	Answer	*BASMATI UI and APIs for users are simple and well defined. It follows the usage procedures of BAMASTI platform.* *Because it is well defined and simple, we had been able to use the APIs without big problems.*
	Error Protection	Does the system prevent incorrect parameter values from being specified? Does the system timely identify any incompatible configurations that would prevent the correct deployment/operation of the cloud application?
	Answer	*If invalid parameters are input, it returns error messages or just ignores the input and does nothing. However, the error messages can be improved to describe the problem properly.*
	User interface design	Is the user interface pleasing? Is it well designed?
	Answer	*UI is not optimal one, but it is working and helpful. It is difficult to use if the user isn't used to the design and procedures of BASMATI.*

Reliability	Availability	Please report on the conducted monitoring.
	Answer	*We have deployed and tested MVD application on BASMATI for several months.* *Before we used to the BEAM and BASAMATI APIs, there were many struggles. However, after we used to that, we have experienced little problems with BASMATI itself.*
	Fault tolerance	Did you experience downtime of the platform during your tests? If yes how many times? How many errors were raised without affecting your experience with the platform? If yes, please describe the errors. How many fatal errors were raised?
	Answer	*Since each components are managed by different entities in different time zones, from time to time, some components were not available online.* *In that case, the application can't be deployed or managed even if it is running on clouds.* *Otherwise, we could deploy and manage our applications without severe problems.*
	Recoverability	Please report on the mean time to recover BASMATI as well as to recover your application (components).
	Answer	*Because most basmati components are using docker container technology and it supports auto-recovery feature, the down time is short enough not to cause critial issues.* *Some fault cases of MVD application were recovered within 10 minutes, while redeployment were required to recover for several other cases.*
Security	Data storage	Is it possible to access the application data? What kind of security mechanisms are in place in order to secure your applications data? Can your own custom security mechanisms be use to secure our application?
	Answer	*For MVD use case, we are depending on ssh for security.* *Even user generated data is not protected by encryption technology, since it is managed separately from golden disk image, we can add some protection using 3rd party technologies like encrypted file systems or external secure storages.* *If an application owns its data as files, they can use same protection technologies.*
	Datatransport	Which mechanisms are in place to secure the data transport (a) between your application components and (b) between your components and BASMATI?
	Answer	*MVD uses scp for transferring data between Hosts.*

		And, it uses HTTP authentication to access REST API of BASAMTI platform.
	System administration	Which mechanisms are in place to securely access (a) BASMATI provisioned VMs and (b) the BASMATI administration UI?
	Answer	*Users can use provided ssh RSA key to access all his/her VMs.* *BASMATI administrators can access the workspace using ID and password.*
Maintainability	Analysability	Can errors and bugs easily be traced? Are sufficient logs provided?
	Answer	*The number of error messages and logs are not sufficient for debugging.* *They need to be more specific too.*
	Modifiability	Is the source code available for everyone? Can bugfixes be committed by everyone? If the answer is "no": Who is in charge of doing so?
	Answer	*All source codes are being managed in BASMATI project's own git server and only accessible by BASMATI members.* *Maybe it can be open to public after this project's period ends.*
	Testability	Can the whole BASMATI framework be tested or single components? Is it clear how to test the components? Please report on the test coverage if possible.
	Answer	*After all BASMATI components has been tested independently, then we has integrated component-to-component.* *A deliverable is describing the procedures of individual tests and integration.*
Overall	Ease of Use	Are the components easy to install, configure, and use?
	Answer	*If user references the deliverables of BASMATI, he/she can deploy, configure the components and use APIs of them without problems.*
	Usefulness	Were you able to operate your application with the BASMATI services?
	Answer	*Yes*
	Documentation	Is a documentation of all BASMATI components the user need to interact with available? Do you consider the documentation useful?
	Answer	*Actually, there are some deliverables that describe APIs and installation procedures of BASMATI components.*

		And, I think they are helpful for users who wants to use BASMATI platform.

8.3 Questionnaires and Interview Results from CAS

8.3.1 First interview

Criteria	Attribute	Questions
Functinal suitabiliy	Completeness	Does the BASMATI system adequately cover the needs of the application? Does it accommodate the platform requirements of the application (e.g. operating system, Java environment, etc.)?
	Answer	*BASMATI even over-covers the needs of the DAS FEST Application. Only part of the BASMATI functionality was used. Yes, the DAS FEST application requirements have been fulfilled.*
	Correctness	*In this pilot demonstrator we had an issue with the bandwidth as it was not able to transmit that much data. So the configuration of when and how the data is transmitted was adapted during runtime. It´s hard to get to a proper setting beforehand. Therefore, the system may behave incorrectly when adapting the setting. Hard to tell if is was behaving correctly at the end.*
Performance	Time behavior	How long does it take to deploy a normal mid-size application (e.g. having 2 components running on two VMs)? Are scale-in and scale-out events handled in time so that a good time behavior of the deployed application is guaranteed?
	Answer	*Assuming you already have the tosca application description and BASMATI is already set up, it takes approximately half an hour including the check that is work properly.*
	Resource utilization	How much hardware (harddisk space, RAM, CPU) are needed for the whole BASMATI framework itself to run?
	Answer	*Hard disk is not an issue. For a clean BASMATI system; A regular server with 32 GB RAM and 8 cores would be satisfactory. Depending on how you use BASMATI, and to which extend you run single components, the demand can grow concerning storage and CPU (e.g. if you run the advances machine learning algorithms in the knowledge extractor, you may have the need to add more CPU and storage.*
Compatibility	Interoperability	Does BASMATI support the deployment of any application with any OS and hardware requirements? Which restrictions are already known?

	Answer	*The most important server operating systems like Unix-based and Windows are supported. Concrete restrictions are not known to me.*
	Modularity	Can different modules of BASMATI be used in isolation?
	Answer	*Yes. They are all independent WS.*
	Adaptability	Can single modules of BASMATI be adapted? Are extension points provided to do so? Is the source code well-documented or self-explanatory?
	Answer	*Yes, but it´s limited. There are not specific plug-n-play modules. But the source code is open source. The component´s code that I was in touch with was self-explanatory.*
Usability	Recognisability	Do the component interfaces follow standard patterns?
	Answer	*All components expose REST interfaces and the interaction between them is well designed.*
	Learnability	Are the interfaces (UI and API) laid out logically, allowing the system to be learned quickly?
	Answer	*The higher level of the API is logical. E.g. to submit an application description. Overall the API is complex in order to fulfil all possible requirements – it´s a general-purpose system.*
	Error Protection	Does the system prevent incorrect parameter values from being specified? Does the system timely identify any incompatible configurations that would prevent the correct deployment/operation of the cloud application?
	Answer	*No, the system does not prevent from entering incorrect values, but it identifies syntactically incorrect application descriptions once submitted. But this process is not really user friendly.*
	User interface design	Is the user interface pleasing? Is it well designed?
	Answer	*ACE´s interface is functional but not too pleasing. The general BASMATI entry point interface is nice and easy to understand and use.*
Reliability	Availability	Please report on the conducted monitoring.
	Answer	*Hard to say. The only downtime we had was due to the fact that the satellite connection was a bottleneck. We adapted the size of the to-be-transported data packages and the time interval to submit them.*
	Fault tolerance	Did you experience downtime of the platform during your tests? If yes how many times? How many errors were raised without affecting your experience with

		the platform? If yes, please describe the errors. How many fatal errors were raised?
	Answer	*The platform was stopped several times during DAS FEST to solve some problems related to the application. During the running period, no fatal error from the platform were raised.*
	Recoverability	Please report on the mean time to recover BASMATI as well as to recover your application (components).
	Answer	*The platform can be restarted in few minutes. In case of DAS FEST, the recover time of the whole application was few (<10) minutes as well.*
Security	Data storage	Is it possible to access the application data? What kind of security mechanisms are in place in order to secure your applications data?
	Answer	*I do not have the necessary knowledge to answer regarding this aspect.*
	Datatransport	Which mechanisms are in place to secure the data transport (a) between your application components and (b) between your components and BASMATI?
	Answer	*Communication between application components and Basmati components happens on REST over HTTPS. For the BASMATI platform, the design also specifies the usage of authorization certificates for the components, in order to identify that the requester is a valid BASMATI component.*
	System administration	Which mechanisms are in place to securely access (a) BASMATI provisioned VMs and (b) the BASMATI administration UI?
	Answer	*I do not have the necessary knowledge to answer regarding this aspect.*
Maintainability	Analysability	Can errors and bugs easily be traced? Are sufficient logs provided?
	Answer	*It is not easy to traces bugs, as essentially each component logs independently. Once logs are found, they are enough to understand the problem.*
	Modifiability	Is the source code available for everyone? Can bugfixes be committed by everyone? If the answer is "no": Who is in charge of doing so?
	Answer	*As now, the source code is available to the member of the BASMATI consortium. [note: I guess that at the end of the project the codebase will be make publicly available, and people have the possibility to submit bug report.]*
	Testability	Can the whole BASMATI framework be tested or single components? Is it clear how to test the

		components? Please report on the test coverage if possible.
	Answer	*I think the whole BASMATI platform can be tested in principle, but the testing of individual components seems much more simple.*
Overall	Ease of Use	Are the components easy to install, configure, and use?
	Answer	*Yes, most of the components are encapsulated in docker container, which extremely simplify their management*
	Usefulness	Were you able to operate your application with the BASMATI services?
	Answer	*Yes, the DAS FEST application was running smooth toward the end of the festival.*
	Documentation	Is a documentation of all BASMATI components the user need to interact with available? Do you consider the documentation useful?
	Answer	*There's not a full BASMATI manual, but all the pieces of documentation are scattered in the various deliverables.*

8.3.2 Second interview

Criteria	Attribute	Questions
Functinal suitabiliy	Completeness	Does the system adequately cover the needs of the application? Does it accommodate the platform requirements of the application (e.g. operating system, Java environment, etc.)?
	Answer	*Yes. All application requirements have been accommodated.*
	Correctness	Does the BASMATI system behave as expected? Does it behave correctly?
	Answer	*Yes.*
Performance	Time behavior	How long does it take to deploy a normal mid-size application (e.g. having 2 comonents running on two VMs)? Are scale-in and scale-out events handled in time so that a good time behavior of the deployed application is guaranteed?
	Answer	*6 minutes if the application description is already available. For creating the application description in tosca, it depends on the complexity of the application software, for the DAS FEST application is was half an hour.*
	Resource utilization	How much hardware (harddisk space, RAM, CPU) are needed for the whole BASMATI framework itself to run?
	Answer	*For the BASMATI framework as we used it on-site at DAS FEST, it was 8GB RAM, 4 Cores, 1TB disk (not used completely), the cloud it was 4GB, 2 cores, 40GB disk.*
Compatibility	Interoperability	Does BASMATI support the deployment of any application with any OS and hardware requirements? Which restrictions are already known?
	Answer	*BASMATI allows deployment of both unix-based and Windows applications.*
	Modularity	Can different modules of BASMATI be used in isolation?
	Answer	*Yes. The BASMATI front end could also be used to generate tosca application descriptions together with another cloud management backend, such as cloudify.*
	Adaptability	Can single modules of BASMATI be adapted? Are extension points provided to do so? Is the source code well-documented or self-explanatory?
	Answer	*Yes, e.g. the application controller has been adapted to satisfy the special needs of the DAS FEST*

		application components.
Usability	Recognisability	Do the component interfaces follow standard patterns?
	Answer	*The component interfaces are based on open standards – including WS-agreement, OCCI and tosca.*
	Learnability	Are the interfaces (UI and API) laid out logically, allowing the system to be learned quickly?
	Answer	*It depends of the background knowledge. It is a powerful platform requiring domain knowledge.*
	Error Protection	Does the system prevent incorrect parameter values from being specified? Does the system timely identify any incompatible configurations that would prevent the correct deployment/operation of the cloud application?
	Answer	*No. The current system is a prototype and would require industrialization.*
	User interface design	Is the user interface pleasing? Is it well designed?
	Answer	*The platform provides technical user interfaces and is intended to be interfaced with business process management interfaces.*
Reliability	Availability	Did you experience downtime of the platform during your tests? If yes how many times?
	Answer	*The platform was operational during the whole DAS FEST setup. The only downtime observed was due to external connectivity issues (satellite disconnection and raspberry pi power offs).*
	Fault tolerance	How many errors were raised without affecting your experience with the platform? If yes, please describe the errors. How many fatal errors were raised?
	Answer	*No fatal error.*
	Recoverability	Please report on the mean time to recover BASMATI as well as to recover your application (components).
	Answer	*Recovering from scratch of BASMATi takes 5 minutes. Recovering all the DAS FEST application components took approx. 30 minutes.*
Security	Data storage	Is it possible to access the application data? What kind of security mechanisms are in place in order to secure your applications data?
	Answer	*Concerning the BASMATI platform data, it is protected by an ORBAC security system requiring authentication and authorization of bath users and software agents. The application data is stored in a Mongo DB without further mechanisms since the data is pseudonymized anyway.*

	Datatransport	Which mechanisms are in place to secure the data transport (a) between your application components and (b) between your components and BASMATI?
	Answer	*All platform communication is through TLS1.2 using pke and x501 certificates. Communication between application components was secured in a similar manner.*
	System administration	Which mechanisms are in place to securely access (a) BASMATI provisioned VMs and (b) the BASMATI administration UI?
	Answer	*Access to VMs is only possible via SSH using SSH keys (no username/passwords)*
Maintainability	Analysability	Can errors and bugs easily be traced? Are sufficient logs provided?
	Answer	*All requests and responses between components are logged.*
	Modifiability	Is the source code available for everyone? Can bugfixes be committed by everyone? If the answer is "no": Who is in charge of doing so?
	Answer	*Interface source code is fully available on git lab.*
	Testability	Can the whole BASMATI framework be tested or single components? Is it clear how to test the components? Please report on the test coverage if possible.
	Answer	*Functional confidence tests are available. The test coverage is component-dependant.*
Overall	Ease of Use	Are the components easy to install, configure, and use?
	Answer	*The ease of use is also component-dependant. In general, it´s easy to use for someone with domain-specific knowledge.*
	Usefulness	Were you able to operate your application with the BASMATI services?
	Answer	*Yes.*
	Documentation	Is a documentation of all BASMATI components the user need to interact with available? Do you consider the documentation useful?
	Answer	*Documentation is available.*

8.3.3 Third interview

Criteria	Attribute	Questions
Functional suitability	Completeness	Does the system adequately cover the needs of the application? Does it accommodate the platform requirements of the application (e.g. operating system, Java environment, etc.)?
	Answer	*Yes. The use case application needs have been satisfied and all base-line requirements like resource use optimization or federation with edge components has been achieved.*
	Correctness	Does the BASMATI system behave as expected? Does it behave correctly?
	Answer	*Yes.*
Performance	Time behavior	How long does it take to deploy a normal mid-size application (e.g. having 2 components running on two VMs)? Are scale-in and scale-out events handled in time so that a good time behavior of the deployed application is guaranteed?
	Answer	*Assuming that the application description and the containers for the application components are available, it takes approx. 5 minutes. SLA violation based scale-out was not necessary for the DAS FEST but could be handled in time.*
	Resource utilization	How much hardware (harddisk space, RAM, CPU) are needed for the whole BASMATI framework itself to run?
	Answer	*Don´t know.*
Compatibility	Interoperability	Does BASMATI support the deployment of any application with any OS and hardware requirements? Which restrictions are already known?
	Answer	*As long as the application is containerized or some other sort of automatic (bash script based) deployment is provided, there are no restrictions known.*
	Modularity	Can different modules of BASMATI be used in isolation?
	Answer	*Yes. Single components of BASMATI can be replaced by custom components. The interaction between the components is based on OCCI and it is straight forward to add new components.*
	Adaptability	Can single modules of BASMATI be adapted? Are extension points provided to do so? Is the source code well-documented or self-explanatory?

	Answer	*Some components like e.g. the knowledge extractor can and should be adapted to dedicated application requirements.*
Usability	Recognisability	Do the component interfaces follow standard patterns?
	Answer	*Yes, we rely on the OCCI standard.*
	Learnability	Are the interfaces (UI and API) laid out logically, allowing the system to be learned quickly?
	Answer	*It depends on the component. The dashboard for BASMATI is very nice and is laid out logically. Regarding the APIs, it is relatively easy to get used to the BASMATI APIs relying on the OCCI standard.*
	Error Protection	Does the system prevent incorrect parameter values from being specified? Does the system timely identify any incompatible configurations that would prevent the correct deployment/operation of the cloud application?
	Answer	*No.*
	User interface design	Is the user interface pleasing? Is it well designed?
	Answer	*It depends on the component. The dashboard for BASMATI is very pleasing. The ACE UI is functional but a bit cumbersome.*
Reliability	Availability	Did you experience downtime of the platform during your tests? If yes how many times?
	Answer	*For the DAS FEST is was 100% available, the only shutdown was because of a satellite disconnection. Other problems we encountered were due to the data format and encryption but this did not affect BASMATI, but rather the application that was deployed via BASMATI.*
	Fault tolerance	How many errors were raised without affecting your experience with the platform? If yes, please describe the errors. How many fatal errors were raised?
	Answer	*The problems that happened during the DAS FEST had nothing to do with BASMATI itself, rather one application component has not been that fault tolerant. During the project, fault tolerance was not the number one priority in the component development.*
	Recoverability	Please report on the mean time to recover BASMATI as well as to recover your application (components).
	Answer	*Does not take long and is easy to do.*
Security	Data storage	Is it possible to access the application data? What kind of security mechanisms are in place in order to secure

		your applications data?
	Answer	*Via Budamaf there are mechanisms to isolate and secure the one application´s data from another. Standard, trustworthy security mechanisms are in place.*
	Datatransport	Which mechanisms are in place to secure the data transport (a) between the BASMATI components and (b) between your components and BASMATI?
	Answer	*Secure transport protocols are in place for most of the BASMATI components.*
	System administration	Which mechanisms are in place to securely access (a) BASMATI provisioned VMs and (b) the BASMATI administration UI?
	Answer	*BASMATI admin UI is username and password restricted. Login on the VMs is done via SSH and keys.*
Maintainability	Analysability	Can errors and bugs easily be traced? Are sufficient logs provided?
	Answer	*Yes.*
	Modifiability	Is the source code available for everyone? Can bugfixes be committed by everyone? If the answer is "no": Who is in charge of doing so?
	Answer	*Most of the components code will be made publically available.*
	Testability	Can the whole BASMATI framework be tested or single components? Is it clear how to test the components? Please report on the test coverage if possible.
	Answer	*Single components can be tested. The test coverage is different from component to component.*
Overall	Ease of Use	Are the components easy to install, configure, and use?
	Answer	*Yes.*
	Usefulness	Were you able to operate your application with the BASMATI services?
	Answer	*Yes.*
	Documentation	Is a documentation of all BASMATI components the user need to interact with available? Do you consider the documentation useful?
	Answer	*Yes.*

8.4 Questionnaires and Interview Results from CNR

8.4.1 First interview

Criteria	Attribute	Questions
Functinal suitabiliy	Completeness	Does the system adequately cover the needs of the application? Does it accommodate the platform requirements of the application (e.g. operating system, Java environment, etc.)?
	Answer	*Yes, all the virtual resources that compose the TripBuilder use case cloud-based service can be accommodated through BASMATI. BASMATI is able to configure different Operating Systems (Linux, Windows) on top of the target resources used, as well as support different application language environments.*
	Correctness	Does the BASMATI system behave as expected? Does it behave correctly?
	Answer	*Yes, we were able to specify the resources needed through a BEAM document, which includes a TOSCA descriptor defining the amount of resources needed by the TripBuilder service. The specification provided allow us to define which parts of the service are fixed and which ones can growth dynamically. Adding load balancers, proxies or other auxiliary application components to the TripBuilder online server can exploit off-the-shelf software components, by deploying them as VMs.*
Performance	Time behavior	How long does it take to deploy a normal mid-size application (e.g. having 2 comonents running on two VMs)? Are scale-in and scale-out events handled in time so that a good time behavior of the deployed application is guaranteed?
	Answer	*Assuming that BEAM description for the application is available, it takes approx. 5 minutes. SLA violation based scale-in/scale-out was configured to raise penalties if the application detects that the execution time for each request exceeds certain threshold.*
	Resource utilization	How much hardware (harddisk space, RAM, CPU) are needed for the whole BASMATI framework itself to run?
	Answer	*In terms of HW, 8GB RAM, 4 Cores, 1TB disk is the minimal capacity required to run an on-site BASMATI*

		framework. However, BASMATI can be configured with a variable set of components, so the amount of resources needed may vary depending on the application requirements.
Compatibility	Interoperability	Does BASMATI support the deployment of any application with any OS and hardware requirements? Which restrictions are already known?
	Answer	*BASMATI allows deployment of both unix-based and Windows applications.*
	Modularity	Can different modules of BASMATI be used in isolation?
	Answer	*Yes, BASMATI framework has been conceived to be modular, current BASMATI component can be replaced by custom components. In this case the new components must be compliant with the OCCI interactions defined for the component, in order to maintain consistency across the whole framework.*
	Adaptability	Can single modules of BASMATI be adapted? Are extension points provided to do so? Is the source code well-documented or self-explanatory?
	Answer	*Yes, single modules of BASMATI can be modified or adapted as long as the inbound/outbound interfaces defined remains unaltered. Most of the modules that compose the framework are licensed under open-source which plays in favor of future adaptations that may be requested.*
Usability	Recognisability	Do the component interfaces follow standard patterns?
	Answer	*All components expose well documented REST interfaces and the interaction between them rely on OCCI standard interfaces that we made available among components via a Publisher module. The designs of the exchanged metadata, as well as the design of our interfaces, consider already well-known standards including WS-Agreement, TOSCA or the aforementioned OCCI.*
	Learnability	Are the interfaces (UI and API) laid out logically, allowing the system to be learned quickly?
	Answer	*It depends on the component, it is a powerful platform requiring domain knowledge for certain operations.* *From the user perspective, in general, REST and OCCI interfaces provided by the components are well defined and in most cases self-explanatory.* *BASMATI UI is simple an intuitive. The definition of BEAM descriptors may require certain degree of*

		knowledge but it relys on well-known standards (TOSCA, WS-Agreement) which should help to decrease the learning curve.
	Error Protection	Does the system prevent incorrect parameter values from being specified? Does the system timely identify any incompatible configurations that would prevent the correct deployment/operation of the cloud application?
	Answer	*No. The current system is a prototype and requires better error handling mechanism.* *In case of incorrect values, the interfaces raise an error but in case of incorrect values within the description of the services it only identifies syntactically incorrect applications description once submitted.*
	User interface design	Is the user interface pleasing? Is it well designed?
	Answer	*Several components of the framework provide their own UI Access endpoints. The different UIs are integrated within a pleasant BASMATI dashboard that is modular, well designed and easy to use. However, the framework offers mainly technical user interfaces that can be difficult to use if the user has no experience with the design and procedures of BASMATI.*
Reliability	Availability	Please report on the conducted monitoring.
	Answer	*Health checks of running components and resources can be performed requesting this information to the Publisher interface as well as relying on the monitoring information gathered from the resources.*
	Fault tolerance	Did you experience downtime of the platform during your tests? If yes how many times? How many errors were raised without affecting your experience with the platform? If yes, please describe the errors. How many fatal errors were raised?
	Answer	*During the implementation/testing phases we detected downtime of some of the components due to the servers where those components were deployed were infected. After applying the remedial actions and restart the servers no further incidents were detected.*
	Recoverability	Please report on the mean time to recover BASMATI as well as to recover your application (components).
	Answer	*Recovering from scratch of BASMATI takes around 5~10 minutes. However, after the attack to some servers, the restarting process included the cause*

		analysis, remedial actions to prevent it to happen again and the redeployment/reconfiguration of the virtual resources. That whole process took around 1 week.
Security	Data storage	Is it possible to access the application data? What kind of security mechanisms are in place in order to secure your applications data? Can your own custom security mechanisms be use to secure our application?
	Answer	*In the cases where BuDaMaf service is used, there are mechanisms to isolate and secure the one application ´s data from another. Standard, trustworthy security mechanisms are in place.* *Overall, the BASMATI platform data, it is protected by an ORBAC security system requiring authentication and authorization of bath users and software agents. Security concerns have been analysed and discussed as part of D5.3 & D.5.5.*
	Datatransport	Which mechanisms are in place to secure the data transport (a) between your application components and (b) between your components and BASMATI?
	Answer	*All platform communication is through TLS1.2 using PKE and x501 certificates.*
	System administration	Which mechanisms are in place to securely access (a) BASMATI provisioned VMs and (b) the BASMATI administration UI?
	Answer	*a) VMs are only accessible via SSH keys certificate.* *b) UI administration is accessible via user/password.*
Maintainability	Analysability	Can errors and bugs easily be traced? Are sufficient logs provided?
	Answer	*All requests and responses between components are logged. In addition, each component provides their own logging mechanism, despite this, due to the complexity of the flow in some cases is difficult to trace the root causes of the error for a proper error handling.*
	Modifiability	Is the source code available for everyone? Can bugfixes be committed by everyone? If the answer is "no": Who is in charge of doing so?
	Answer	*Interface source code is fully available on BASMATI git lab. However, bugfixes during the project lifetime have been solved by the component owners according to the continuous integration procedures specified in the WP2.*

	Testability	Can the whole BASMATI framework be tested or single components? Is it clear how to test the components? Please report on the test coverage if possible.
	Answer	*Functional tests are provided within maven projects used to build the components. Test-coverage is component dependant.*
Overall	Ease of Use	Are the components easy to install, configure, and use?
	Answer	*The ease of use is dependent on the number of modules involved for your applications, in some cases certain degree of domain-specific knowledge is required.*
	Usefulness	Were you able to operate your application with the BASMATI services?
	Answer	*Yes.*
	Documentation	Is a documentation of all BASMATI components the user need to interact with available? Do you consider the documentation useful?
	Answer	*Yes.*

8.4.2 Second interview

Criteria	Attribute	Questions
Functinal suitabiliy	Completeness	Does the system adequately cover the needs of the application? Does it accommodate the platform requirements of the application (e.g. operating system, Java environment, etc.)?
	Answer	*Yes, the BASMATI platform supports the requirements of the application.*
	Correctness	Does the BASMATI system behave as expected? Does it behave correctly?
	Answer	*Yes, the application behaves as intended.*
Performance	Time behavior	How long does it take to deploy a normal mid-size application (e.g. having 2 components running on two VMs)? Are scale-in and scale-out events handled in time so that a good time behavior of the deployed application is guaranteed?
	Answer	*Few minutes. If application instances are already instantiated, the scaling up/down works in acceptable time limits.*
	Resource utilization	How much hardware (harddisk space, RAM, CPU) are needed for the whole BASMATI framework itself to run?
	Answer	*Regular workstation parameters, such as 8/16 GB of RAM, 4/8 CPU cores, > 1TB of disk space. SSD can improve performances.*
Compatibility	Interoperability	Does BASMATI support the deployment of any application with any OS and hardware requirements? Which restrictions are already known?
	Answer	*If properly configured, BASMATI can run on any major operative systems.*
	Modularity	Can different modules of BASMATI be used in isolation?
	Answer	*BASMATI is a collection of services, that can be used in isolation if needed.*
	Adaptability	Can single modules of BASMATI be adapted? Are extension points provided to do so? Is the source code well-documented or self-explanatory?
	Answer	*All the BASMATI components are open-source, so they can be modified with some domain Knoledge. The level of documentation is component dependent.*
Usability	Recognisability	Do the component interfaces follow standard patterns?

	Answer	*Yes, all components uses REST interfaces and JSON payloads.*
	Learnability	Are the interfaces (UI and API) laid out logically, allowing the system to be learned quickly?
	Answer	*Out-of-the-box operation are simple, for more complex usage domain knowledge is required.*
	Error Protection	Does the system prevent incorrect parameter values from being specified? Does the system timely identify any incompatible configurations that would prevent the correct deployment/operation of the cloud application?
	Answer	*Not regularly. Some components might have this functionalies, but generally no.*
	User interface design	Is the user interface pleasing? Is it well designed?
	Answer	*There are many interfaces with different level of polishing and fucntionalities. Overall the level of interfaces is acceptable.*
Reliability	Availability	Please report on the conducted monitoring.
	Answer	*We monitored the response time of the application instance over multiple requrets.*
	Fault tolerance	Did you experience downtime of the platform during your tests? If yes how many times? How many errors were raised without affecting your experience with the platform? If yes, please describe the errors. How many fatal errors were raised?
	Answer	*Minor downtime events due to regular restarting of the application (mostly due to testing of new features).*
	Recoverability	Please report on the mean time to recover BASMATI as well as to recover your application (components).
	Answer	*BASMATI takes few minutes to be recovered, as well as the application.*
Security	Data storage	Is it possible to access the application data? What kind of security mechanisms are in place in order to secure your applications data? Can your own custom security mechanisms be use to secure our application?
	Answer	*Tripbuilder does not have particularly sensible data to protect. To access the data an attacker would need to enter the VM instance, so the level of security depends on the isolation supported by the cloud service provider, and on the configuration of the VM.*
	Datatransport	Which mechanisms are in place to secure the data transport (a) between your application components and (b) between your components and BASMATI?

	Answer	*Communications in the BASMATI platform support HTTPS.*
	System administration	Which mechanisms are in place to securely access (a) BASMATI provisioned VMs and (b) the BASMATI administration UI?
	Answer	*VM are protected and normally accessed with SSH keys. Also, that depends on the isolation level provided by the cloud service provider in which the VM is running.*
Maintainability	Analysability	Can errors and bugs easily be traced? Are sufficient logs provided?
	Answer	*Errors are logged by each component. It can be difficult to spot an error are log are not kept in the same place.*
	Modifiability	Is the source code available for everyone? Can bugfixes be committed by everyone? If the answer is "no": Who is in charge of doing so?
	Answer	*Code is open source. Bug reporting feature was open to the consortium only during the project.*
	Testability	Can the whole BASMATI framework be tested or single components? Is it clear how to test the components? Please report on the test coverage if possible.
	Answer	*Some components can be tested.*
Overall	Ease of Use	Are the components easy to install, configure, and use?
	Answer	*Most of components use Docker, which simplifies their deployment and execution.*
	Usefulness	Were you able to operate your application with the BASMATI services?
	Answer	*Yes.*
	Documentation	Is a documentation of all BASMATI components the user need to interact with available? Do you consider the documentation useful?
	Answer	*Yes, documentation is useful.*

References

9 M. Hegner, Methoden zur Evaluation von Software, IZ Sozialwissenschaften, 2003.

[2] D. M. a. G. R. José P. Miquel, "A Review of Software Quality Models for the Evaluation of Software Products," *International Journal of Software Engeneering and Applications ,* pp. Vol. 5, No. 6, 2014.

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

[4] T. Koponen, "Evaluation Framework for Open Source Software Maintenance," *International Conference on Software Engineering Advances,* p. IEEE, 2006.

[5] R. Dromey, "A model for software product quality," *IEEE Transactions on Software Engineering ,* no. 21, pp. 146-162, 1995.

[6] "ISO/IEC 25010:2011 (en) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models," ISO, 2011. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:35733:en. [Accessed 01 08 2016].

[7] F. AlQayedi, K. Salah and M. J. Zemerly, "Adaptive Cloud Resource ALlocation schema to minimize SLO response time violation," *Computer Systems and Applications,* vol. 13th International Conference of IEEE/ACS, 2016.

[8] S. Moreno, P. Garraghan, P. Townend and J. Xu, "An approach for characterizing workloads in google cloud to derive realistic resource utilization models," *Service Oriented Engineering (SOSE) ,* vol. 7th IEEE International Symposium, 2013.