Grant Agreement Number: 257528

KHRESMOI

www.khresmoi.eu

Prototype and Evaluation of the 'Full KHRESMOI Integrated Infrastructure'

Deliverable number	*D6.5.3*
Dissemination level	*Public*
Delivery date	*August 2014*
Status	*Final*
Author(s)	*Ivan Martinez (AtoS), Miguel Angel Tinte (AtoS)*

This project is supported by the European Commission under the Information and Communication Technologies (ICT) Theme of the 7th Framework Programme for Research and Technological Development.

Table of Contents

List of figures

List of tables

List of abbreviations

SOA	Service Oriented Architecture
SCA	Service Component Architecture
SOMA	Service-Oriented Modelling Architecture
REST	Representational State Transfer
API	Application Programming Interface
eZDL	Easy Access to Digital Libraries
NC	Network Cohesion
NSIC	Number of Services Involved in a Compound Service
AIS	Absolute Importance of a Service
ADS	Absolute Dependence of a Service
ACS	Absolute Criticality of a Service
DoW	Description of Work
VM	Virtual Machine
RPC	Remote Procedure Call
SSH	Secure Shell
PING	Packet Internet Groper
HTTP	Hypertext Transfer Protocol
Mb	Megabytes
Gb	Gigabytes

1 Executive summary

This document describes the evaluation performed over the "Full KHRESMOI Integrated Infrastructure", as an iterative improvement process which started with the “Early Integrated Infrastructure” prototype and continues after a refinement phase with the deployment of the final prototype. The current final stage represents as previously the result of joining the “Full Software architecture” deployed within the “Full Cloud Prototype” to achieve scalability, flexibility and elasticity requirements as well as the last improvements came up after the end-users evaluation process performed during the last year of the project. Hence, this document describes mainly the work achieved in Tasks “T6.4 System integration”, concretely T6.4.4 [10] and T6.4.5 [11] in charge of refinement and evaluation respectively. Following the same approach than for previous deliverables D6.5.2 [1], this document describes the evaluation processes done re-using previous evaluation metrics and results reported in deliverables D6.4.2 [6] and D6.4.3 [3]. In the same way, this document describes the same scenario-independent tests and different related scenarios tests which contain some changes and improvements that will be fully described. The main outcome in this context will be comparing results between both prototypes when possible, taking into account the new improvements developed and assessing their impact in the performance of the system as a whole. These conclusions will be highly useful in order to face the end of the project and define a strategy of items achieved or remaining in order to deploy a production-like environment.

Apart from this main empirical evaluation work, the document will take into account the feedback from reviewers and end-users as a complementary evaluation in order to cover all different types of requirements expected to be supported on the Full KHRESMOI Integrated infrastructure.

Finally, this final report aims to show an overall view of the KHRESMOI final infrastructure that reflects clearly all the work performed by all the partners in terms of integration, usability and scalability of the platform.

2 Introduction

2.1 Introductory Explanation of the Deliverable

The main objective of this document is to show a final overview of the "Full KHRESMOI Integrated infrastructure" as the final step of the KHRESMOI platform development and deployment. This final version combines the technical requirements described in the DoW [9] in terms of Software components integration and performance scalability as well as final refinements coming from the last user evaluations performed during the last year. The description of the "Full Integrated Infrastructure" is showed along with an evaluation chapter where different tests performed are described to assess the infrastructure. These tests results allow us to use some predefined metrics to evaluate numerically some performance features.

Aligned to this approach, the document will describe all the improvements and refinements that affect to the platform compared with the previous prototypes, in terms of components, scenarios, metric, etc.

2.2 Purpose and Audience

2.2.1 Purpose

The purpose of this deliverable is to describe the Full KHRESMOI Integrated infrastructure where the main improvements and refinements updated over previous prototype are described as the result of Tasks [10] and [11].

2.2.2 Audience

Following the same approach than in previous occasions for D6.5.2 [1], D6.3.3 [4], this deliverable is relevant to all technical work packages in KHRESMOI (WP1-WP9). The target audience includes component providers, users, and any person inside or outside of the KHRESMOI project interested in learning about the internal processing of the KHRESMOI software architecture. As such this deliverable presents a description of the KHRESMOI Full Prototype software architecture, and therefore this document could be interesting for any KHRESMOI member. Besides that, this document acquires an important relevance to the rest of the partners due to it represents a final assessment report from the KHRESMOI final platform.

Because the document is public it is also intended for anybody interested in the evaluation and integration of KHRESMOI.

2.3 Structure of the Document

Similarly to previous evaluations deliverables, this document re-uses the same structure of the previous specification deliverable D6.5.2 [1], D6.5.1 [2] and includes new sections in cases where new functionalities, metrics or scenarios have been included. The list of sections defined in the document are the following: Section 2 focuses on describing the Global Evaluation Approach adopted in order to evaluate architecture and cloud solutions developed, Section 3 defines the test scenarios definition integrated so far into different workflows. Section 5 focuses on describing scenario independent results and in Section 6 the scenario dependent result along with the main performance metrics is described. This document includes also the Section 7 to explain the improvements or refinements

adopted after some feedback received from projects reviewers as well as some requirements extracted after performing the end-users test during project last year. Finally, Section 8 is dedicated to describe the main conclusions obtained after evaluation process.

3 Global Evaluation Approach

The global evaluation approach is the same as in previous deliverables D.6.5.2 [1]. Similarly to previous evaluation, empirical assessment will be divided into two different blocks, one regarding architecture evaluation and another one regarding cloud infrastructure.

In this occasion, another important outcome of the evaluation will be the comparison between both prototypes assessments as well as the description of the last functionalities refinements and end-users recommendations.

3.1 Architecture Evaluation Approach

The architecture evaluation approach is already defined in D6.5.2 [1].

3.2 Cloud Infrastructure Evaluation Approach

The cloud infrastructure evaluation approach is already defined in D6.5.2 [1].

4 Tests Scenario Definition

This Chapter describes different workflow scenarios simulated in order to obtain different results and metrics to assess the main integration and scalability concepts. All of them have been already simulated in previous evaluations to assess each infrastructure deployment, so it is referenced in each case:

4.1 Textual Search Scenario

This scenario has been already described in previous deliverables D6.5.2 [1] and D6.4.3 [3]. The main update in this workflow in terms of new functionalities is the inclusion of different facets related to medical concepts, in the results retrieved. These facets are retrieved requesting to Mimir RPC service and they are included into the results in order to enrich the output offered to end-user. Table below shows the workflow components and methods where the last updates appear highlighted:

Prototype 1	**Textual search**	
Components	ezDL (UDE)	ezDL is a multi-agent search system for heterogeneous data sources and a tool-set for building search user interfaces to support complex tasks
	Speller (HON)	HON's medical multilingual spell checking service
	Multilingual Translator (CUNI)	CUNI Multilingual translator service
	Disambiguator (ONTO)	ONTO Disambiguation service
	KMI (USFD)	KMI Service allows an XML file will be posted to the service, which will result in a Mimir search result being returned. The response from this service will be the same as a standard `postQuery` call to the Mimir web service API.
	Mimir (USFD)	Mimir search Web Service
	MimirFacetedSearch (AtoS)	**Implementation of Mimir RPC service to retrieve documents facets**
	SCA TextualSearchManager (AtoS)	This component manages the complete Textual Search Workflow that integrates the other components and its iterations
Scenario	Step1 : search(keywords)	The user introduces a list of keywords through the UI in order to obtain a proper answer. In this version, the system allows also to add some constraints in order to adjust the search.

	Step2 : improve query	The user obtains some improvements to the query keywords performed by the system: spelling correction, possible language translation and disambiguation.
	Step3 : perform search of final query	The query is transformed by KMI Component into a query_id that will be used to perform the search over the index
	Step4 : return list of Documents	The user receives a list of documents with hits found as the answer of the search
	Step5 : view document	The user can see and translate any document of the list returned from the search
Main functionalities to be integrated	Functionalities used by User: from ezDL	search(List<Keywords>)
	Functionalities used by User: from ezDL	viewDoc(docURI)
	Functionalities used by ezDL: from Speller	List<Suggestion> :getSpelling(keywords,lang)
	Functionalities used by ezDL: from MT	translateQuery(text,docType,sourceLang,targetLang,profile)
	Functionalities used by ezDL: from MT	translateDoc(docURI)
	Functionalities used by ezDL: from Disambiguation	List<Label> :getDissambiguation(keywords)
	Functionalities used by ezDL: from TextManager	searchByText(userProfile,keywords)
	Functionalities used by TextManager: from KMI	query_id :getQueryId(kmi-config,kmi-query)
	Functionalities used by TextManager: from Mimir	hitCount(index_id,query_id) hits(index_id,query_id,start_index,count) docMetadata(index_id,query_id,doc_id) docText(index_id,query_id,rank,term_pos) **getDocumentId(index_id, queryId, index)**

	Functionalities used by TextManager: from MimirFacetedSearch	**postTermsQuery(docIds[])**

Table 1: Textual Search scenario

4.2 2D Image Search Scenario

This scenario has been already described in previous deliverables D6.5.2 [1] and D6.4.3 [3]. The only change adopted in the workflow is the inclusion of "RadioWiki" dataset in the list of dataset available. Hence, no changes in the workflow description have been done.

4.3 3D Image search Scenario

This scenario has been already described in previous deliverables D6.5.2 [1] and D6.4.3 [3].

4.4 Multilingual Search Scenario

This scenario has been already described in previous deliverables D6.5.2 [1] and D6.4.3 [3].

4.5 2D Semantic Image Search

This scenario has been already described in previous deliverable D6.5.2 [1]. The main update included in this release is the inclusion of facets from different categories supported (Modalities, Anatomies and Pathologies) as well as adding relevant labels associated to these facets into the results retrieved.

Table 2 below shows all the workflow components and methods involved redefined since last description. Also changes and updates adopted during last year refinement appear highlighted in black:

	2D Semantic Image search	
Components	ezDL (UDE)	The same definition than for D6.5.2 [4]: ezDL is a multi-agent search system for heterogeneous data sources and a tool-set for building search user interfaces to support complex tasks
	ParaDISE (HES)	The same definition than for D6.5.2 [1]: ParaDISE (the GNU Image-Finding Tool) is a Content Based Image Retrieval System.
	SCA-ParaDISE (AtoS)	SCA Component for Multilingual Translator

	Co-occurrence (ONTO)	Co-occurrence Search: A query searches within all fields defined for each entity. The result is a list, which can be further narrowed by using the filter facets.
	SCA Cooccurrence (AtoS)	SCA Component for Cooccurrence
	Dissambiguator (ONTO)	ONTO Dissambiguator service
	SCA Dissambiguator (AtoS)	SCA Component for Dissambiguator
	SCA SemanticSearchManager (AtoS)	This component manages the complete Semantic Search Workflow that integrates the other components and its iterations
Scenario	Step1 : load terms	The user performs an empty query to obtain a list of preselected terms.
	Step2 : parse co-occurrence results	**Terms are disambiguated by SCA Dissambiguator**
	Step 2.1: collect facet URIs with count for categories Anatomy, Pathology and Modalities	**Facets are collected and divided according to cooccurrenceStatistics value and labels from disambiguation are included by SCA SemanticSearchManager**
	Step 3: collect image results	**Images are retrieved by SCA ParaDISE**
	Step 3.1: add image details	**Image details are added into different image result by SCA SemanticSearchManager**
	Step 4: Disambiguate facets URIs	Call Disambiguator service with the collected facet URIs
	Step 5 : response is returned to ezDL	**Returns a list of found images with details from Paradise and list of facets for each category (Anatomy, Pathology and Modalities). Each facet should contain the following information: URI, label and count.**
Main functionalities to be integrated	Functionalities used by User: From ezDL	searchImages()
	Functionalities used by ezDL:	List<PreselectedTerms> = Load()

	from SemanticManager	
	Functionalities used by ezDL: from SemanticManager	**ImagesDetails images = findImages(cooccurrenceStatistics, resultType, currentPage, count, radlexTerms, imagesMap)**
	Functionalities used by Semantic Manager: from SCA Cooccurrence	**Terms = getCooccurrence(radlexTerms, resultType, count, currentPage, cooccurrenceStatistics)**
	Functionalities used by Semantic Manager: from SCA-Disambiguator	List<DisambiguatedTerms> facets = DisambiguateDetails()
	Functionalities used by Semantic Manager: From SCA ParaDISE	List<ImageInfo> = GetImageInfo(List<ImageIDs>)

Table 2: 2D Semantic Image Search scenario

5 Scenario Independent Results

Following the same approach than in previous evaluations, Figure 1 shows all the nodes and components involved in the 'Full KHRESMOI Integrated prototype' where basically it is displayed graphically the different Full Cloud nodes where the SCA Components and Composites have been deployed and the interrelations among them. The updated relations, components or methods are highlighted in black over the Figure:

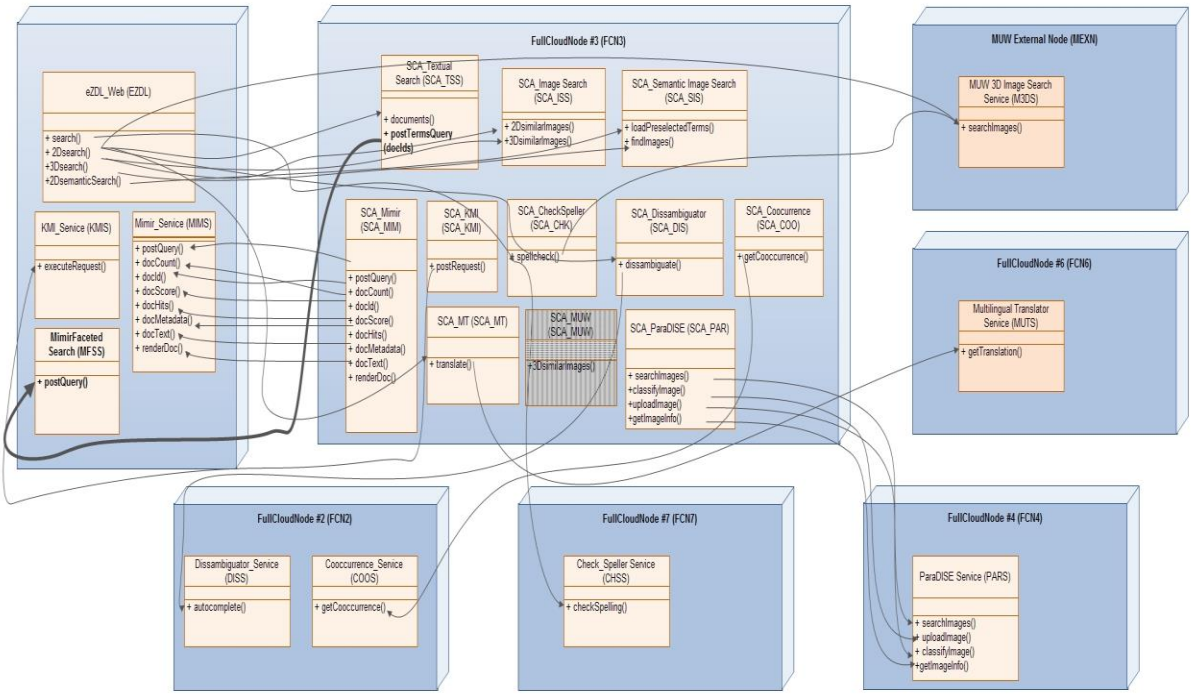


Figure 1: Full KHRESMOI Integrated infrastructure

The following box shows the updated set A of calls between components (this set was already defined in deliverables D6.5.2 [1] and D6.3.3 [5]). The new calls appear highlighted in black:

A = {< FCN1, EZDL, FCN3, SCA_TSS, SCA_TSS.documents()>,
< FCN1, EZDL, FCN3, SCA_ISS, SCA_ISS.2DsimilarImages()>,
< FCN1, EZDL, MEXN, M3DS, M3DS.3DsimilarImages()>,
< FCN1, EZDL, FCN3, SCA_SIS, SCA_SIS.loadPreselectedTerms()>,
< FCN1, EZDL, FCN3, SCA_SIS, SCA_SIS.findImages()>,
< FCN1, EZDL, FCN3, SCA_CHK, SCA_CHK.checkSpelling()>,
< FCN1, EZDL, FCN3, SCA_MT, SCA_MT.translate()>,
< FCN1, EZDL, FCN3, SCA_DIS, SCA_DIS.dissambiguate ()>,
< FCN3, SCA_MIM, FCN1, MIMS, MIMS.postQuery()>,
< FCN3, SCA_MIM, FCN1, MIMS, MIMS.docCount()>,
< FCN3, SCA_MIM, FCN1, MIMS, MIMS.docId()>,

< FCN3, SCA_MIM, FCN1, MIMS, MIMS.docScore()>,

< FCN3, SCA_MIM, FCN1, MIMS, MIMS.docHits()>,

< FCN3, SCA_MIM, FCN1, MIMS, MIMS.docMetadata()>,

< FCN3, SCA_MIM, FCN1, MIMS, MIMS.renderDocument()>,

< FCN3, SCA_KMI, FCN1, MIMS, KMIS.executeRequest()>,

< FCN3, SCA_TSS, FCN1, RPC_MFS, MFS.postTermsQuery()>,

< FCN3, SCA_CHK, FCN7, CHSS, CHSS.checkSpelling()>,

< FCN3, SCA_DIS, FCN2, DISS, DISS.autocomplete()>,

< FCN3, SCA_COO, FCN2, COOS, COOS.getCooccurrence()>,

< FCN3, SCA_MT, FCN6, MUTS, MUTS.translate()>,

< FCN3, SCA_PAR, FCN4, PARS, PARS.searchImages()>,

< FCN3, SCA_PAR, FCN4, PARS, PARS.updateImage()>,

< FCN3, SCA_PAR, FCN4, PARS, PARS.classifyImage()>,

< FCN3, SCA_PAR, FCN4, PARS, PARS.getImagesInfo()>}

5.1 Network Cohesion

Network cohesion metric has been used during previous evaluation processes and its formal definition appears in deliverable [5]. It can be calculated as follows:

$NC = |\pi_{InvokerNode,ServiceNode}(\sigma_{InvokerNode \neq ServiceNode}(A))| = \{<FCN1, FCN3>, <FCN1, MEXN>, <FCN3, FCN1>, <FCN3,FCN2>, <FCN3,FCN7>, <FCN3,FCN4>, <FCN3,FCN6>, <FCN3,MEXN>\} = \mathbf{8}$

The result above obtained for network cohesion metric (8) is the same than the calculated for 'Early Integrated infrastructure' evaluation [1]. This is due to that the numbers of nodes involved in the platform remain stable since last prototype, so therefore, cohesion value remains also stable. Not even the inclusion of a new method call (MFS.postTermsQuery()) modifies the cohesion value. Consequently, the stability in the tests performed over the infrastructure suggests that the network cohesion is quite mature and has reached an optimal degree of integration.

5.2 Number of Services Involved in a Compound Service

This metric has been already defined [5] and used previously in deliverables [1] [2] [3]. The Full KHRESMOI infrastructure supports the same compound service that in previous prototype, concretely SCA_TSS, SCA_ISS, SCA_SIS and ezDL. The metric values for these cases are calculated similarly than for Early Integrated Prototype evaluation. Basically, the assessments retrieve the same values except in those cases where new components or services have been included into the workflow:

$NSIC[c] = |S^R[c]|$

$NSIC[SCA\_TSS] = |S^R[SCA\_TSS]| = \{SCA\_KMI, SCA\_MIM, KMIS, MIMS, RPC\_MFS\} = \mathbf{5}$

$NSIC[SCA\_ISS] = |S^R[SCA\_ISS]| = \{SCA\_PAR, , PARS\} = \mathbf{2}$

$NSIC[SCA\_SIS] = |S^R[SCA\_SIS]| = \{SCA\_COO, SCA\_DIS, SCA\_PAR, COOS, MUWS, PARS\} = \mathbf{6}$

*NSIC[ezDL] = $|S^R[EZDL]|$={SCA_TSS, SCA_ISS, SCA_SIS, SCA_KMI, SCA_MIM, SCA_COO, SCA_DIS, SCA_MT, SCA_MT, SCA_PAR, KMIS, MIMS, MUTS, COOS, DISS, PARS, M3DS}=**17***

Similarly to previous evaluation [1], the results for NSIC metrics show that the SCA Composites have a high value due to the complexity of the workflows implemented. However, the highest value is for still for ezDL as it is the access layer for the end-users. In this last iteration, it is observed that SCA_TSS is the only one that increases its NSIC value because of new faceting feature included. Despite the last refinements and updates, the stability of the rest of NSIC values unveils a mature integration status among the components.

5.3 Services Interdependence in the System

This metric has been already defined in deliverable [5]. As in the previous evaluation, there is no service interdependence in the Full KHRESMOI Integrated prototype so there is not a bidirectional relation between services.

5.4 Absolute Importance of a Service

Absolute importance of a service (AIS) has been already defined [5] and calculated [1][2] in previous evaluations. The results of the evaluation of current prototype are the next:

$$AIS[s] = |\pi_{Invoker}(\sigma_{InvokerNode=s,\ ServiceNode\neq n}(A))|$$

*AIS[ezDL]=**0***

*AIS[SCA_TSS]=**1***

*AIS[SCA_ISS]=**1***

*AIS[SCA_SIS]=**1***

*AIS[SCA_CHK]=**1***

*AIS[SCA_DIS]=**2***

*AIS[SCA_COO]=**2***

*AIS[SCA_KMI]=**1***

*AIS[SCA_MIM]=**1***

*AIS[SCA_PAR]=**2***

*AIS[SCA_MT]=**1***

*AIS[DISS]=**1***

*AIS[CHSS]=**1***

*AIS[KMIS]=**1***

*AIS[MIMS]=**1***

*AIS[MUWS]=**1***

*AIS[PARS]=**1***

*AIS[MUTS]=**1***

*AIS[MFSS]=**1***

The AIS metric can be summarized basically as the number of external components which depend directly on one service. According to the SCA implementation proposed since the beginning of the

integration, this abstraction layer creates at least one direct dependency among services and their SCA implementations. Besides this, SCA Composites could aggregate several components so therefore some of them could relate more than once with another components. Despite this inherent interdependence, all AIS values are below or equal to 2 which can be considered as an acceptable value.

5.5 Absolute Dependence of a Service

For the calculation of this metric (ADS) the definition provided in [5] is taken as reference, which can be calculated as follows:

$ADS[s] = |\pi_{Service}(\sigma_{Invoker=s,\ ServiceNode \neq n}(A))|.$

*ADS[ezDL]=**8***

*ADS[SCA_TSS]=**3***

*ADS[SCA_ISS]=**1***

*ADS[SCA_SIS]=**3***

*ADS[SCA_CHK]=**1***

*ADS[SCA_DIS]=**1***

*ADS[SCA_COO]=**1***

*ADS[SCA_KMI]=**1***

*ADS[SCA_MIM]=**1***

*ADS[SCA_PAR]=**1***

*ADS[SCA_MT]=**1***

*ADS[DISS]=**0***

*ADS[COOS]=**0***

*ADS[CHKS]=**0***

*ADS[KMIS]=**0***

*ADS[MIMS]=**0***

*ADS[MUWS]=**0***

*ADS[PARS]=**0***

*ADS[MTS]=**0***

*ADS[MFSS]=**0***

As explained in deliverable [1], this metric can be defined as the opposite to the previous one, so this shows the level of dependence from this service to the other ones. Equally to previous evaluations, this value gives an idea about the autonomy of the services and possibility of being reused as stand-alone instances where autonomy and reusability are affected by this value. Similarly to previous evaluations, ADS [ezDL] shows the higher value and represents the most important dependence of the infrastructure. Also, ADS[SCA_TSS] has increased due to the incorporation of MFSS in the workflow.

5.6 Absolute Criticality of a Service

This metric has been already defined in previous evaluation deliverables [5]. Basically, it is calculated multiplying AIS by ADS for every service:

ACS[*s*] = *AIS*[*s*] × *ADS*[*s*].

ACS[*ezDL*] = *AIS*[ezDL] × *ADS*[ezDL]=**0**

ACS[*SCA_TSS*] = *AIS*[SCA_TSS] × *ADS*[SCA_TSS]=**3**

ACS[*SCA_ISS*] = *AIS*[SCA_ISS] × *ADS*[SCA_ISS]=**1**

ACS[*SCA_SIS*] = *AIS*[SCA_SIS] × *ADS*[SCA_SIS]=**3**

ACS[*SCA_CHK*] = *AIS*[SCA_CHK] × *ADS*[SCA_CHK]=**1**

ACS[*SCA_DIS*] = *AIS*[SCA_DIS] × *ADS*[SCA_DIS]=**2**

ACS[*SCA_COO*] = *AIS*[SCA_COO] × *ADS*[SCA_COO]=**2**

ACS[*SCA_KMI*] = *AIS*[SCA_KMI] × *ADS*[SCA_KMI]=**1**

ACS[*SCA_MIM*] = *AIS*[SCA_MIM] × *ADS*[SCA_MIM]=**2**

ACS[*SCA_PAR*] = *AIS*[SCA_PAR] × *ADS*[SCA_PAR]=**2**

ACS[*SCA_MT*] = *AIS*[SCA_MT] × *ADS*[SCA_MT]=**1**

ACS[*DISS*] = *AIS*[DISS] × *ADS*[DISS]=**0**

ACS[COOS] = *AIS*[COOS] × *ADS*[COOS]=**0**

ACS[*SPES*] = *AIS*[CHKS] × *ADS*[CHKS]=**0**

ACS[QMSS] = *AIS*[KMIS] × *ADS*[KMIS]=**0**

ACS[*MIMS*] = *AIS*[MIMS] × *ADS*[MIMS]=**0**

ACS[*PARS*] = *AIS*[PARS] × *ADS*[PARS]=**0**

ACS[*MTS*] = *AIS*[MTS] × *ADS*[MTS]=**0**

ACS[*MFSS*] = *AIS*[*MFSS*] × *ADS*[*MFSS*]=**0**

Results above do not show important changes compared with previous evaluations. The new component included has an ACS[MFSS] value of 0 and it affects in another already existing component metric value like ACS[SCA_TSS]=3, which has increased because it includes a dependency with MFSS. Also, ACS[SCA_COO] has increased from to 3 due to its inclusion in Semantic Search workflow, which has been redefined for this prototype. Table 3: below shows a classification of different services criticality results:

Metrics	**Criticality**		
	Very critical (>1)	Critical(=1)	Not critical(=0)
ACS[ezDL]			×
ACS[SCA_TSS]	×		
ACS[SCA_ISS]		×	
ACS[SCA_SIS]	×		

ACS[SCA_CHK]		×	
ACS[SCA_DIS]	×		
ACS[SCA_COO]	×		
ACS[SCA_KMI]		×	
ACS[SCA_MIM]	×		
ACS[SCA_PAR]	×		
ACS[SCA_MT]		×	
ACS[DISS]			×
ACS[COOS]			×
ACS[SPES]			×
ACS[QMSS]			×
ACS[PARS]			×
ACS[MTS]			×
ACS[MFSS]			×

Table 3: Absolute criticality of a service matrix

In comparison with 'Early Integration prototype evaluation', the results in Table above display a very similar values, pointing again to complex workflows as the most likely components to have critical values due to their high interrelation with other services. The main classification of critical services is:

- SCA_TSS, SCA_SIS, SCA_ISS, SCA_MIM, SCA_PAR, SCA_COO with higher value than 1.
- SCA_ISS, SCA_CHK, SCA_KMI, SCA_MT with value equals to 1.

5.7 Overall Reliability of a Compound Service

This metric has been already defined in previous deliverables [5] and it can be calculated using the same formula than for previous evaluations, as follows:

$RC[c] = 1/\max(\{ACS[s] \mid s \in S^R[c]\})$.

$RC[\text{ezDL}] = 1/\max(\{ACS[s] \mid s \in S^R[\text{ezDL}]\}) = 1/3 = \mathbf{0.33}$

$RC[\text{SCA\_TSS}] = 1/\max(\{ACS[s] \mid s \in S^R[\text{SCA\_TSS}]\}) = 1/2 = \mathbf{0.5}$

$RC[\text{SCA\_ISS}] = 1/\max(\{ACS[s] \mid s \in S^R[\text{SCA\_ISS}]\}) = 1/2 = \mathbf{0.5}$

$RC[\text{SCA\_SIS}] = 1/\max(\{ACS[s] \mid s \in S^R[\text{SCA\_ISS}]\}) = 1/2 = \mathbf{0.5}$

Despite the final results remain the same in Early Prototype and in current Full Prototype some difference are remarkable. For instance, ACS[s] has increased for some of the services "s" included in compound services "c" but RC[c] returns the same value because it is only affected by the maximum value of all the ACS[s] involved. Therefore, the maximum is this case is not higher that previous evaluation. RC metric values of all the Compounds Services are stable compared to previous prototype.

6 Scenario Dependent Results and Performance Metrics.

This chapter is focused on the last 'Scenario Dependent Simulations Evaluations' of the project. Hence, these evaluations are not conceived as isolated tests but they are conceived as the final step of the complex evaluation process achieved during previous deliverables [1] [2]. To do so, the evaluation approach has been adopted using the same configuration parameters have been chosen for simulations: 100 users + 4 queries/user + 5 minutes time. The final goal will be to check the evolution of system performance over time, taking into account the changes and updates of the scenarios during project development.

6.1 Textual Search Scenario.

This scenario has suffered relevant changes since the previous Scenario Dependent Simulation evaluations in deliverables [1] [2] [5]. The most relevant updates are related to the inclusion of the facets in the final results previously commented on chapter Textual Search Scenario Also, the inclusion of new parameter configuration allows to the user setting a more accurate configuration of the execution. The main updates are the following:

- The new pagination solution allow to the user the configuration of the results expected, as well as the division of these results into different pages retrieved. To do so, there are three different parameters required *numResults, numPage, sizePage*. Some execution examples would be:
  - *&numPage=1&sizePage=10&numResults=10*
  - *&numPage=1&sizePage=20&numResults=40*
  - *&numPage=2&sizePage=20&numResults=100*
  - Etc.
- The new parameter *metadataFields* allows to the user the configuration of the different metadata values required in each query execution. This feature provides to the system a high flexibility because it supports variable complex queries depending of the numbers of metadata fields requested. Example:
  - *&metadataFields=date*
  - *&metadataFields=date,uri,corpus,publisher,domainClasses,authors*
  - *&metadataFields=* //empty input value retrieves all metadata fields by default
  - Etc.

During the Textual Search scenario evaluation described below, the metrics and results exposed have been obtained with the next Textual Search scenario configuration:

- The first page of 20 results (for a total of 50 results) are requested: *&numPage=1&sizePage=20&numResults=50*
- All metadata fields are requested: *&metadataFields=*

6.1.1 Sizes of Input and Output Messages

Similarly to other metrics, this metric has been defined and used in previous deliverables [1] [5]. The main goal of this metric is to quantify the size of input and output messages that participate in each workflow. In this updated prototype, the way to request to individual SCA Components is the same

but the way to request to Textual Search workflow has been widely changed so currently it is necessary to send a complete XML in the POST method instead that a simple word or topic as we did during previous iterations. This modification causes certainly an increase in the size of input messages sent to the server as results show below. Likewise input size, output size is affected also due to the changes and updates implemented on the results retrieved. The updated results in this iteration are:

- AN1.search: XML input query with term 'pneumonia':

```
<?xml version='1.0' encoding='UTF-8'?>
<nodeBool>
	<negated>false</negated>
	<fieldCode>
		<id>-2</id>
		<sortable>false</sortable>
		<searchable>false</searchable>
	</fieldCode>
	<type>AND</type>
	<children>
		<nodeCompare>
			<negated>false</negated>
			<parent reference="../../.." />
			<fieldCode>
				<id>17</id>
				<type>java.lang.String</type>
				<sortable>false</sortable>
				<searchable>true</searchable>
			</fieldCode>
			<predicate>EQ</predicate>
			<matchable class="match">
				<annotations />
				<term>pneumonia</term>
			</matchable>
		</nodeCompare>
		<nodeCompare>
			<negated>false</negated>
			<parent reference="../../.." />
			<fieldCode>
				<id>19</id>
				<type>java.lang.String</type>
				<sortable>false</sortable>
				<searchable>true</searchable>
			</fieldCode>
			<predicate>EQ</predicate>
			<matchable class="match">
				<annotations />
				<term>en</term>
			</matchable>
		</nodeCompare>
	</children>

</nodeBool>
```

The results values for AIMSO/AOMSO metrics obtained with this query are the following:

AIMSO [SCA_TSS.search] = ***798 bytes***
AOMSO [SCA_TSS.search] = ***5,975 bytes***

The metric results exposed above are only a sample of the possible values that sizes of input and output metrics could take in different use cases. The last updates adopted on the Textual Search workflow confer a great flexibility to the service during requesting execution, allowing different size of input queries, depending of the complexity of XML constraints added to the query. This fact also

directly affects to the output size, which can be also limited with the new paging and numResults parameter. Therefore, the results above have been obtained in a very specific context adopted which represents a simple XML query for terms related to 'pneumonia' and requesting for the first page of 20 results. It is observed how the values obtained are similar to previous iterations so it can be interpreted as a sign of stability which should ensure the capacity of the system to deal the different queries that could be sent to the service.

6.1.2 Messages Rates

To measure Message Rate per Node metric (MRN), we have focused on the updated Textual Search service on Node1, which has been updated since the last evaluation, and we have used the same query than for previous metric. The MRN results for this service are the following:

- AN1.search: pneumonia

IMRN [AN1] = OMRAN [AN1] = ***275msg/min***

MRN [AN1] = ***275*** *+* ***275****=* ***550msg/min***

As showed above, MRN has increased for AN1.search where Textual Search service is deployed. This increase is undoubtedly caused by the new parameters to limit the number of results, which allow to the service to do more requests with fewer results per call. . Surely, as numResults parameter becomes higher the MRN value would decrease proportionally. Besides these refinements and improvements, another changes adopted as threading faceting requests in order to improve execution could affect positively to the MRN value for AN1• .

6.1.3 Network Load

Network load metric has been already described in previous iterations and evaluations [5] [1] and becomes more critical as the project progresses. Its importance relies on the fact that an increasing complexity and amount of information exchanged on the platform required mandatory a network bandwidth able to deal with it. Similarly to previous evaluations, to measure the network load it is collected the sizes of input and output messages, only for AN1 in this prototype:

ITN[AN1] = N x (798+5,975) ***= N x 6,773 bytes/call***

OTN[AN1] = N x (5,975+798) ***= N x 6,773 bytes/call***

The results above display a similar value of bytes exchanged in each request execution, only a bit higher due to increase of XML input query. Despite this, the value of N could be higher in this prototype as platform could be used more exhaustively so therefore network load must be checked continuously during all the project lifetime to ensure a proper performance.

6.1.4 CPU Usage

CPU Usage has been measured again for Textual Search workflow in order to observe the influence of the new refinements implemented in the consumption of this resource. Figure 2 below displays the percentage of consumption during scenario simulation execution:

Figure 2: CPU Usage Textual Search

Figure above shows an interval of CPU consumption between 13-38 %, which are not very high values compared to previous evaluation. However, some peaks in the consumption can be observed at some point of the simulation, which can be caused by overlapping of virtual users, concurrent requests, etc. Looking to following diagrams may be useful to obtain extra information about these peaks, so for example it is observed below in Figure 6, around minutes 1' and 2' the concurrent virtual users are up to 3 at some moments.

Anyway, this behavior can be considered as normal for KHRESMOI platform purposes, in order to be accessible simultaneously to several users, so the tests performed so far ensure a successful CPU consumption during execution.

6.1.5 Memory Usage

Another critical hardware resource which affect to system performance is Memory Usage. In this case, Figure 3 below shows a quite stable consumption around 1.68 Gb during whole execution. This value is higher than during previous evaluations [1], what could be caused by increase of workflow complexity that would require also a higher allocation of memory to be able to deal with execution process. Despite this increase, it is observed there are not peaks on consumption as well as atos1 VM still keeps free memory (from its 32 memory allocated Gb) during simulation. Hence, results obtained from Memory Usage can be considered as good during Textual Search simulation execution.

Figure 3: Memory Usage Textual Search

6.1.6 Average Response Time

Average Response Time metric (ART) is one of the most useful metrics to get a general overview of system performance. In this context, 'Full KHRESMOI integrated' prototype represents the last iteration of the platform development that has been achieved during the project where different

workflows implemented aim to fulfill the use cases identified. Aligned with this approach, Textual Search workflow execution is highly dependent on the response time employed on end-users request.

Figure 4 below shows the diagram where ART values along the execution are displayed:

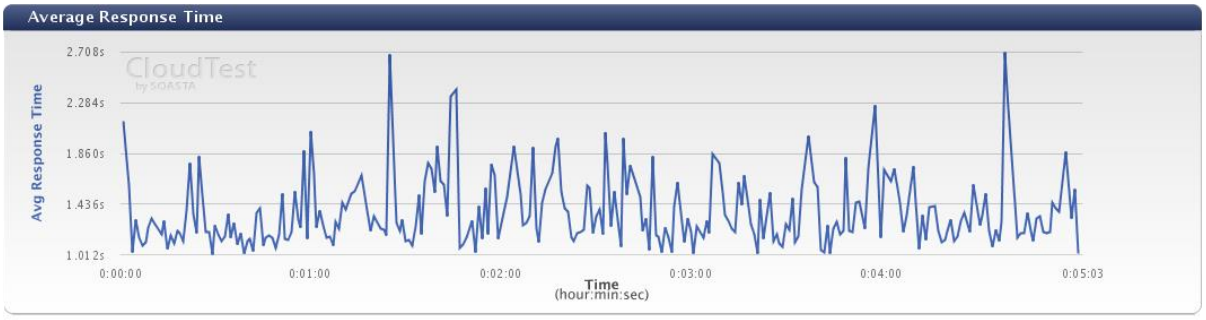


Figure 4: Average Response Time Textual Search

As it can be seen above, three main Response Time thresholds are disguised: the first one until 1.436 seconds include most of the time response registered. The second one appears between 1.436 and 1.860 seconds, where several time responses drawn are included. Finally, it is observed some time records around 2.284 seconds, reaching in some cases values of 2.708 seconds. Taking into account these results it seems quite obvious which performance is not always as good as desirable: all Time Responses higher than 1.5 seconds can be considered as not optimal. In this simulation, we found many results over this limit which could be caused by several reasons including virtual users overlapping, CPU peaks consumption, etc. Besides these physical causes, the updated software refinements adopted during last year of projects (including facets, paging solution, metadata field customization, etc.) increase undoubtedly the complexity of the workflow and directly affects to time execution and times responses.

In any case, 'Full KHRESMOI Integrated Prototype' has showed enough ability to deal with different types of queries within Textual Search workflow with acceptable Times Responses. The less optimal Time Responses detected reflects the complexity of the components integration which directly affects to queries performance. This final evaluation should be taken into account after project end at the time of deployment a production-like environment. Some possible solutions should cover different aspects such as improving hardware resources; optimize component and network integration, etc.

6.1.7 Network Bandwidth

Figure 5 shows the bandwidth usage diagram for Textual Search simulation. The bandwidth consumption is very stable but there are some peaks which could be related to some virtual users overlapping as described above.

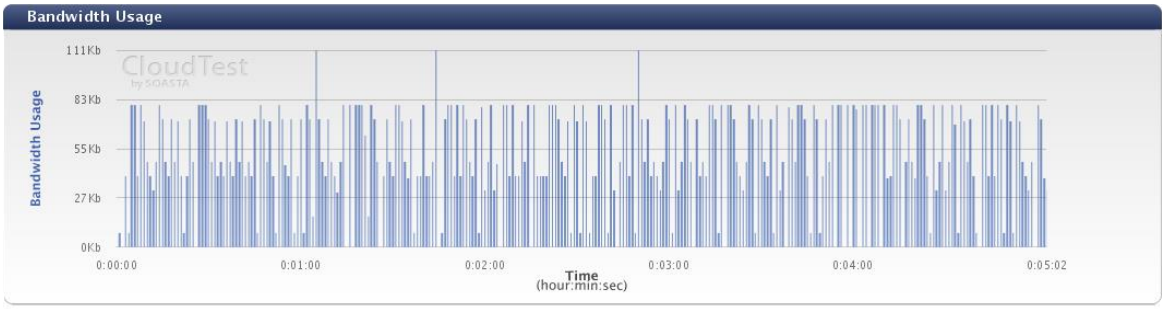


Figure 5: Network bandwidth Textual Search

In general terms, the network bandwidth consumption in shown as optimal to deal with several queries performed over the platform.

6.1.8 Virtual Users

Virtual Users diagram is showed in Figure 6, where it is showed some of the characteristics already commented. Simulation execution has been programmed to emulate 100 users requesting over the platform, with a maximum of 4 queries per user. As can be shown below, users usually tend to overlap their executions due to the large number of requests performed (what is a very likely scenario).

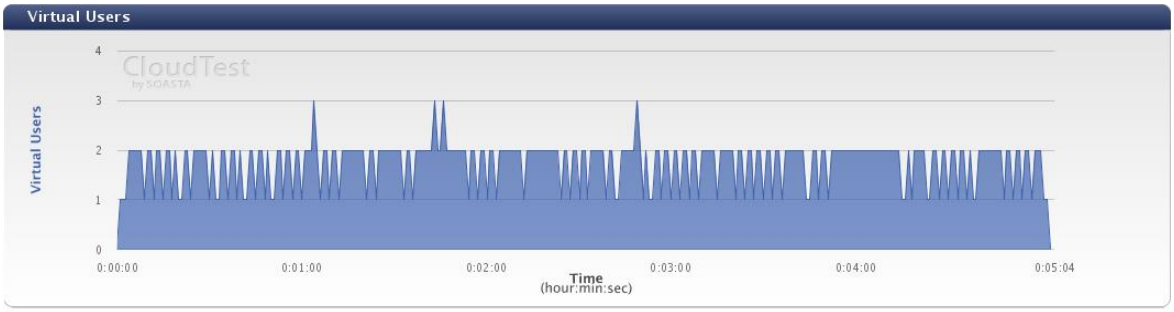


Figure 6: Virtual Users Textual Search

According to this scenario, it is observed which the system is able to deal with several virtual users requesting simultaneously without an excess execution overlapping and avoiding this way the apparition of bottlenecks in the performance.

6.2 2D Image Search Scenario.

This scenario simulation has been already assessed and appears along with its metric results in previous deliverables [1] [3] [4]. In this iteration, 'Full KHRESMOI Integrated prototype' contains the same workflow implementation than the previous prototype so therefore it is not necessary to assess again the 2D Image Search scenario.

6.3 Multilingual Textual Search Scenario.

This scenario is a special case of Textual Search scenario, where input query terms are translated on-the-fly into a different language before performing the search by the system. This use case has been deprecated due to the changes on input query sent to the system, so currently is not possible simulate a Multilingual Textual Search on-the-fly without processing the input XML to extract the terms to be translated. Therefore, this final evaluation document focuses only on results evaluation of Textual Search workflow in chapter 6.1

6.4 2D Semantic Image Search Scenario.

Semantic Image Search workflow has been redefined in this prototype since the last implementation deployed on the 'Early Integrated prototype'. The new workflow has been fully described in chapter 4.5. The main changes in Semantic Search queries concern the inclusion of new parameters to add new features to the query:

http://atos1-khresmoi.ms.mff.cuni.cz:8080/khresmoi-semantic-search-dev/rest/terms/search?

- *cooccurrenceStatistics=Anatomies%2CPathologies%2CModalities*

- *&resultType=ImageClef*
- *&radlexTerms=Anatomies%3A%3Ahttp%3A%2F%2Fbioontology.org%2Fprojects%2Fontologies%2Fradlex%2FradlexOwlDlComponent%23RID1968|arm%26q%3DModalities%3A%3Ahttp%3A%2F%2Fbioontology.org%2Fprojects%2Fontologies%2Fradlex%2FradlexOwlDlComponent%23RID10337*
- *&count=100*
- *&currentPage=1*
- *&imagesMap={"http%3A%2F%2Ffast.hevs.ch%2Fimages%2Fimagesets%2Ffull_indexing_pubmed%2Fimages%2Ff5%2F1752-1947-2-154-2.jpg"%3Atrue}*

The main functionalities provided by parameters above are the following:

- cooccurrenceStatistics: It is used to determine the medical concepts used by the query
- resultType: This parameter allows to set the dataset that will be used to perform the query
- radlexTerms: This parameter contains the concrete terms that compose the query. These terms are passed through a concrete format required by co-occurrence service.
- count: It is used to set the number of results to be retrieved.
- currentPage: To choose the page required from the total number of results
- imagesMap: This new parameter included in this iteration allows to pass as input an existing image in the dataset to complement the query

Subsequently, the following subchapters are dedicated to the evaluation of this updated scenario in order to obtain the most relevant metrics results, setting aside those which have not suffered important changes and lack of interest to assess the performance.

6.4.1 Sizes of input and Output Messages

This metric has been measured in several occasions for different scenarios where they have been formally defined [4]. In this occasion, AIMSO[] and AOMSO[] metrics have been measured for the second time within Semantic Search scenario and have focused only on those steps which have suffered changes since last iteration. Therefore, AIMSO[SCA_2DSIS.search] and AOMSO[SCA_2DSIS.search] have been recalculated due to changes suffered on workflow execution for this iteration. Results obtained are displayed below:

- AN1.SIS.search:

*SO [SCA_2DSIS.search] = **449 bytes***
*AOMSO [SCA_2DSIS.search] = **146,381 bytes***

As showed above, both metrics values have increased. Firstly, AIMSO[SCA_2DSIS.search] has lightly increased due to new parameters included in request (which have been described previously) and in second term, AOMSO[SCA_2DSIS.search] has increased seriously its value. This fact can be caused to changes on repository requested, as well as changes included in output returned, such as the inclusion of the labels in the results, etc. In next subchapters it will observed how this increase of results size may affect n the rest of metrics evaluation.

6.4.2 Messages Rates

This metric is focused also on the assessment of the AN1.search method which represents the Semantic Search workflow. The metric results presented below show the Message Rates values obtained after the simulation. The main conclusion regarding these results is the increase of AOMSO value affects to the value of IMRN[AN1] so it is observed a decreasing of MRN value around the 20%:.

- AN1.search :

IMRN [AN1] = OMRAN [AN1] = ***67 msg/min***

MRN [AN1] = 67 + 67 = ***134 msg/min***

The consequences of this effect will be also checked along with the rest of the metrics, such as Time Response, Network load metric and others.

6.4.3 Network Load

As a consequence of the MRN[AN1] changes described above, the ITN metric results displayed below shows higher network bandwidth consumption. The current prototype, 'Full KHRESMOI Integrated infrastructure', consumes almost a 30% more of network bandwidth that for the 'Early Integrated prototype'. This change implies an increasing consumption of physical resources that could affect to the performance of the platform.

ITN [AN1] = N x (449 + ***146,381*** *) = N calls/min x 146830 bytes/call =* ***N x 143 Kbytes/call***

*ITN [AN1] = N x (****146,381 + 449*** *) = N calls/min x 146830 bytes/call =* ***N x 143 Kbytes/call***

6.4.4 CPU Usage

CPU Usage is displayed in Figure 7 with the percentage consumption required during the simulation execution. It is observed how the average consumption oscillates between 6-13 % although some peaks are observed during the execution. These peaks registered reach values up to 27%, which is not a critical value. Therefore, despite the increase of Messages Size and Network bandwidth, the CPU consumption observed does not require reach critical values so it is able to deal with Semantic Search scenario concurrent queries.

Figure 7: CPU Usage Semantic Search

6.4.5 Memory Usage

Figure 8 diagram shows an increasing of Memory Consumption average compared with 'Early Integrated prototype'. The current consumption is stable around 1.68 GB which can be interpreted as a good sign due to the lack of peaks consumption or bottlenecks.

Figure 8: Memory Usage Semantic Search

6.4.6 Average Response Time (ART)

Another metric that could be affected by the increase of MRN [AN1] and ITN [AN1] is Average Response Time (ART). ART metric has been already evaluated in previous deliverables [4]. Its importance relies on the fact that its effects are directly appreciated by end-users whilst others could have long-terms effects not appreciable directly. Figure 9 below displays the ART values registered during Semantic Search simulation:

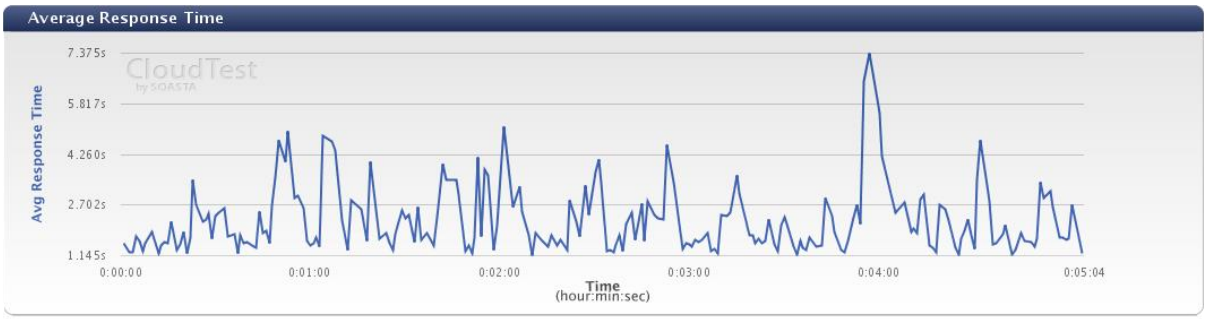


Figure 9: ART Semantic Search

As it can be observed above, ART values are higher than for previous evaluations; most of the values are between 1.145-2.702 seconds interval whilst some values reach values of 4.260 seconds and even 7.375 as the maximum. These results can be considered as not optimal due to ART as higher than 1.5 seconds in several requests. These results can be explained by complexity increase of Semantic Search workflow during last iteration development, as well as size of input and output messages. One possible solution to improve ART would be using the new parameters added to the service in order to perform more accurate queries and therefore reducing MRN and IRN values.

6.4.7 Network Bandwidth

Network bandwidth metric has been assessed along with the rest of the metrics in the previous evaluation deliverables [1] [2]. Similarly to ART metric, Network bandwidth consumption is clearly affected by MRN[AN1] and ITN[AN1] increase. This can be justified because Bandwidth required to deal with requests is directly proportional to AIMSO[AN1] and AOMSO[AN2]. Figure 10 below displays the results obtained during simulation:

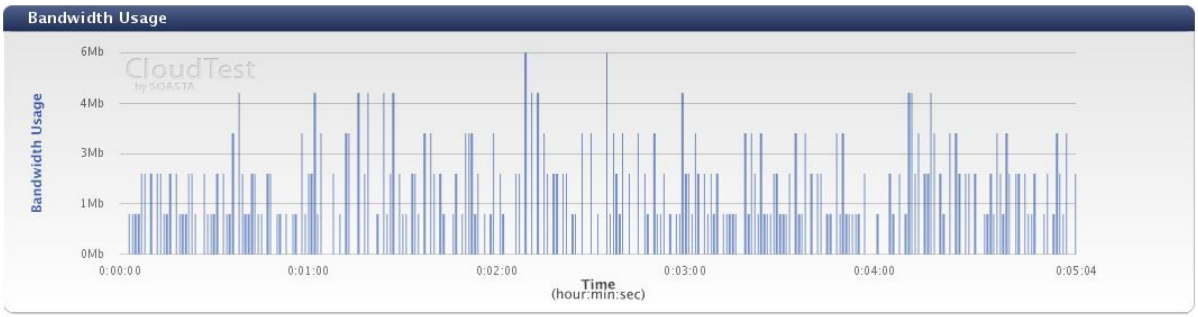


Figure 10: Network bandwidth Semantic Search

The results above show a higher bandwidth usage values than in previous prototype evaluation. Despite these high values are like 10 times bigger now, the simulation displays a quite distributed usage of the bandwidth during all the execution. This means that the system is able to manage all requests and exchange the information required without queuing them and avoiding bottlenecks. From the point of view of bandwidth usage, setting aside the ART results, we can determine the results as positive even though some peaks are detected, probably caused by simultaneous users queries performed. Next, Virtual Users metric results are presented, where some correlations can be found by comparing their results with Network Bandwidth usage.

6.4.8 Virtual Users

Figure 11 below shows the virtual users values reached during the simulation. Regarding previous metrics commented, Virtual Users values acquire more relevance correlated with ART and Bandwidth usage. According to values displayed, Virtual Users show a small overlapping during all the execution, appearing two simultaneous users during most of the time whilst the highest values detected are four simultaneous users and even six simultaneous users as the maximum. These peaks of simultaneous users are directly related with higher ART which appear minute 4' as well as higher bandwidth consumption.

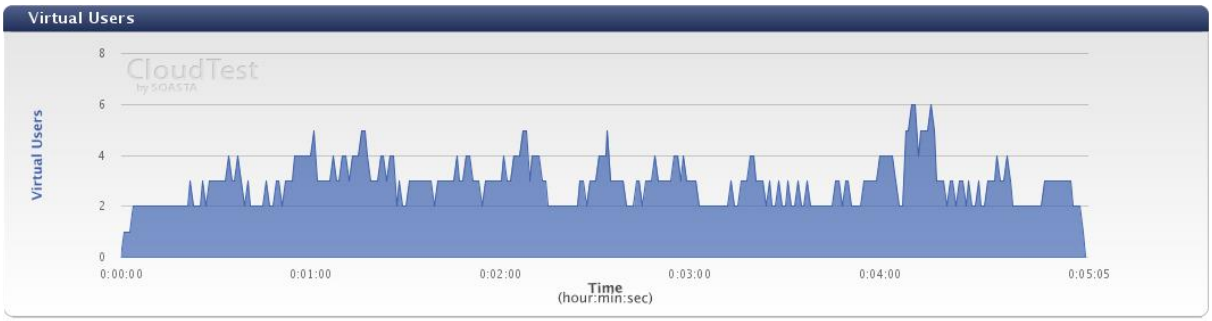


Figure 11: Virtual Users Semantic Search

In general terms, the execution can be considered as optimal, due to the large number of requests performed with different and simultaneous users and delivered successfully without bottlenecks, out of time responses from server, etc.

The most disturbing cases are those in which many users match multiple simultaneous queries affecting both the available bandwidth and the ART. In the future, these cases should be taken into account when deploying a system to meet these functional requirements.

7 Refinements and improvements for final prototype

This chapter focuses on describing specific changes or updates developed and adopted for the 'Full KHRESMOI Integrated prototype'. These changes have been motivated mainly by recommendations from different evaluators, end-users, etc. The two main groups of changes identified come from the project reviewers' suggestions and recommendations from the Third Review Meeting, and from the feedback obtained from different end-users evaluation tests performed during the last year of the project.

7.1 Inclusion of facets

One of the new functionalities included in the Textual Search workflow during last year has been the incorporation of facets in the results retrieved in order to fulfil the demands from the end-users. Facets allow facetted browsing (or some other kind of navigation) within a query result list, presenting clearly distinct tabs for the various facets of a query.

The interface for the medical professionals currently displays the following facets (using data from Mimir documents):

- page category: these are the categories assigned to domains ("domainClasses" metadata, can be multivalued);
- location: the country of origin for the result page ("locationCountry" metadata);
- publisher: either an actual publisher or the domain of the page ("publisher" and "domain" metadata, keep them separated);
- target audience: through the use of the 'hesClassification', even though the values do not seem to fit physicians' expectations of "professional" or "patient" sites ("hesClassification" metadata)
- language: language of the page ("language" metadata)

The example below shows facets fields retrieved within the Textual Search workflow output:

```
<ns2:facet name="hesClassification">
      ...
</ns2:facet>
<ns2:facet name="locationCountry">
 <ns2:value>
      <count>1</count>
      <name>US</name>
 </ns2:value>
</ns2:facet>
<ns2:facet name="language">
      ...
</ns2:facet>
<ns2:facet name="publisher">
      ...
</ns2:facet>
```

7.2 Nagios Network Analyzer monitor service

Nagios Network Analyzer (NagiosNA) [1] is the monitoring software service chosen to check the HTTP traffic over the Full Cloud Network. This feature has been added in this iteration after some comments from the reviewers, (#Recommendation 6) on the KHRESMOI Year 3 review report [8], where they encouraged the project to check how the Full Cloud infrastructure is being used in terms of HTTP traffic requests, with information from IPs addresses from requesters, as well as quantifies the data flow exchanged in bytes, etc. Nagios fits perfectly to this purpose.

Figure 12 below displays the different sources currently monitored with NagiosNA:

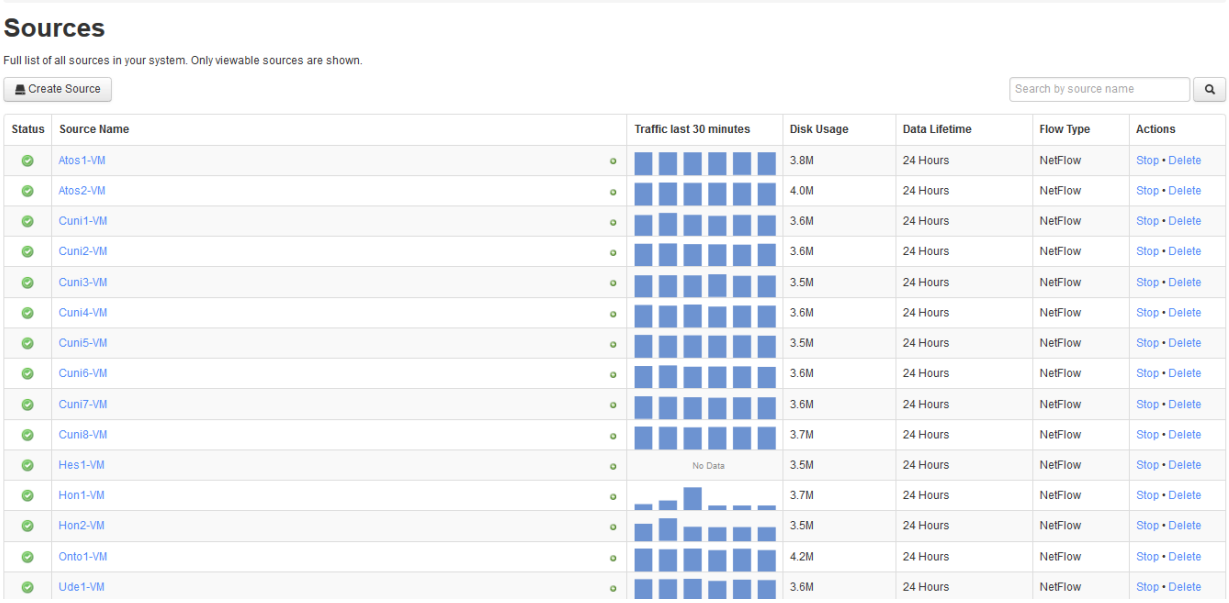


Figure 12: Nagios Network Analyzer Sources

The figure above shows the list of sources monitored with Nagios Network Analyzer. Basically, it is monitoring all the VMs from different partners, where the main information displayed is the traffic during the last 30 minutes, the disk usage in terms of Mb used, the source lifetime period of the data information provided and the flow type of the monitor source.

Figure 13 shows more detailed information related to a specific resource, in this case Atos1-VM, where the SCA Composites are deployed.

[1] http://www.nagios.com/products/nagios-network-analyzer

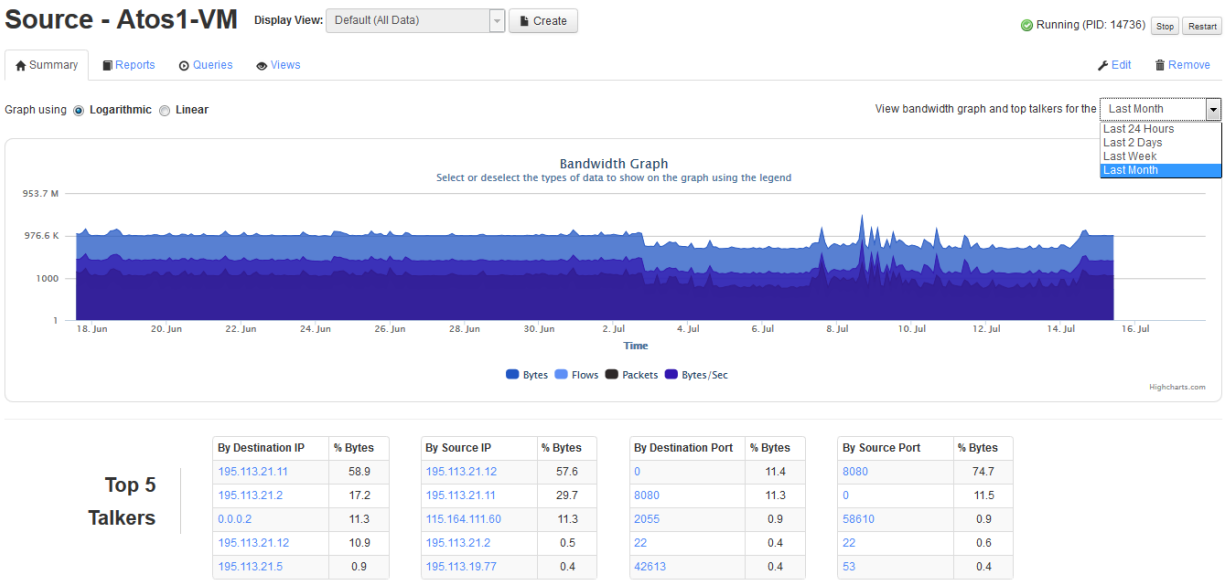
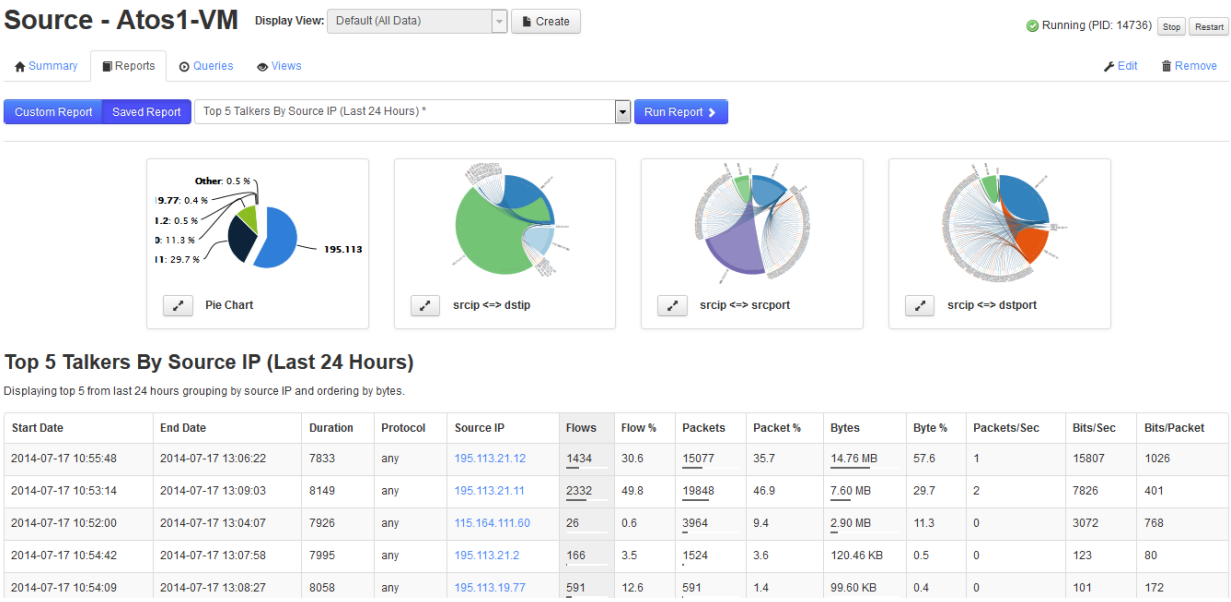


Figure 13: Atos1 VM bandwidth graph

As shown above, the Bandwidth graph displays the bandwidth consumption information during a variable period of time, which can be selected from the list of values which appear in the diagram: last 24 hours, last 2 days, last week and last month. The diagram is able to be rendered showing Bytes, Flows, Packets and/or Bytes/seconds.

Besides this, the diagram displays information about the 'Top 5 talkers', which represent the nodes or VMs that most often communicate with the source selected. These IPs addresses are ranked decreasingly by percentage of bytes exchanged and are divided into two different tables: 'By Destination IP' and 'By Source IP'. Apart from the IPs addresses, two extra tables are included related to the most commonly used ports: 'By Destination Port' and 'By Source port'.

Figure 14 shows another tab included in the source information that creates automatic reports displayed as a different types of diagrams. The first diagram on the left-hand side of the figure is a pie chart about the 'Top 5 talkers IPs', while the rest are chord diagrams representing the following information: 'Source IP to Destiny IP', 'Source IP to Source Port', Source IP to Destiny Port'.



Start Date	End Date	Duration	Protocol	Source IP	Flows	Flow %	Packets	Packet %	Bytes	Byte %	Packets/Sec	Bits/Sec	Bits/Packet
2014-07-17 10:55:48	2014-07-17 13:06:22	7833	any	195.113.21.12	1434	30.6	15077	35.7	14.76 MB	57.6	1	15807	1026
2014-07-17 10:53:14	2014-07-17 13:09:03	8149	any	195.113.21.11	2332	49.8	19848	46.9	7.60 MB	29.7	2	7826	401
2014-07-17 10:52:00	2014-07-17 13:04:07	7926	any	115.164.111.60	26	0.6	3964	9.4	2.90 MB	11.3	0	3072	768
2014-07-17 10:54:42	2014-07-17 13:07:58	7995	any	195.113.21.2	166	3.5	1524	3.6	120.46 KB	0.5	0	123	80
2014-07-17 10:54:09	2014-07-17 13:08:27	8058	any	195.113.19.77	591	12.6	591	1.4	99.60 KB	0.4	0	101	172

Figure 14: Atos1 VM diagram reports

These diagrams provided on the "reports" tab add a great feature to the monitor service allowing to the user a quick overview of the HTTP traffic running over a concrete source. Apart from these visual tools, the "reports" tab also complements 'Top 5 talkers' table with extra information such as: Protocol, Flows, Packets, etc.

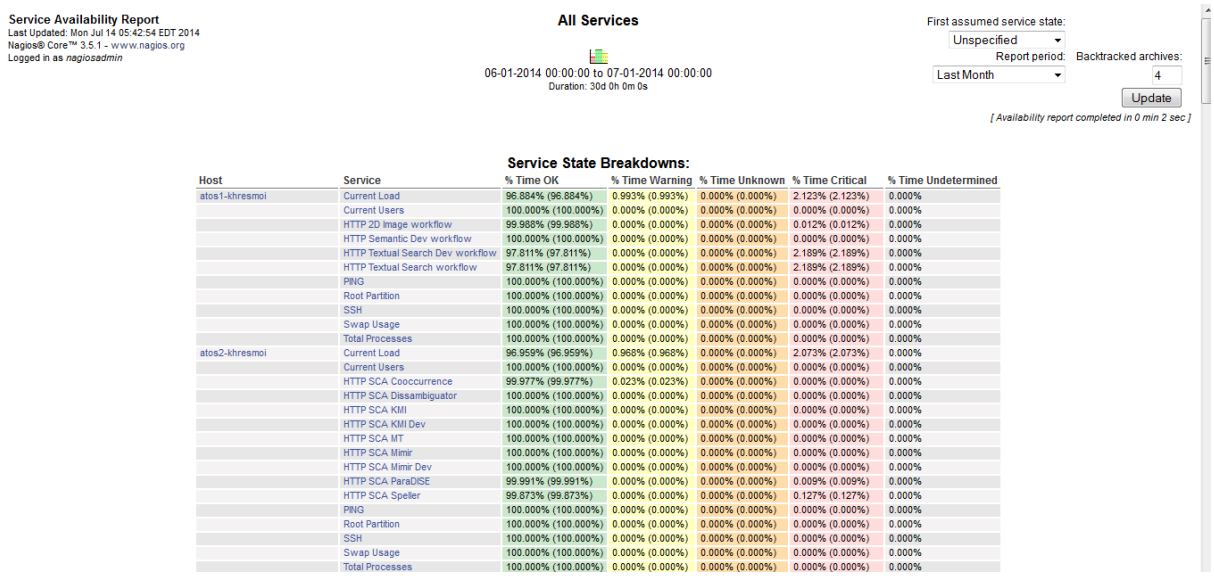
In summary, Nagios Network Analyzer tool has been selected and included in this iteration as an extra monitoring tool to cover the need of checking how the Full Cloud is being used in terms of HTTP data flows. Nagios provides information such as 'who' is using the services as a report of IP addresses, 'which 'services are being used as a report of ports and VMs used, etc. This information plays a key role in order in the assessment of the performance of the Full KHRESMOI Integrated prototype.

7.3 Availability monthly report

The 'Availability Monthly report' aims to create an automatic report of availability of nodes and VMs during last month of operation, and it is directed to all partners who should be informed. This new feature answers to one of the recommendation (#Recommendation 6) on the KHRESMOI Year 3 review report [8]. This report includes information about nodes availability (in percentage of time) as well as availability from all the services hosted on the nodes. The following query invokes a REST API that retrieves a detailed report about KHRESMOI Full Cloud VMs and nodes availability:

http://node0-khresmoi.ms.mff.cuni.cz/nagios/cgi-bin/avail.cgi?t1=1396324800&t2=1396858993&show_log_entries=&host=Host1&service=all&assumeinitialstates=yes&assumestateretention=yes&assumestatesduringnotrunning=yes&includesoftstates=no&initialassumedhoststate=0&initialassumedservicestate=0&timeperiod=lastmonth&backtrack=4

Figure 15 shows the information included and displayed in the monthly availability report, taking Atos1 and Atos 2 VMs as example:



Service Availability Report
Last Updated: Mon Jul 14 05:42:54 EDT 2014
Nagios® Core™ 3.5.1 - www.nagios.org
Logged in as *nagiosadmin*

All Services

06-01-2014 00:00:00 to 07-01-2014 00:00:00
Duration: 30d 0h 0m 0s

First assumed service state: Unspecified
Report period: Last Month
Backtracked archives: 4
Update
[Availability report completed in 0 min 2 sec]

Service State Breakdowns:

Host	Service	% Time OK	% Time Warning	% Time Unknown	% Time Critical	% Time Undetermined
atos1-khresmoi	Current Load	96.884% (96.884%)	0.993% (0.993%)	0.000% (0.000%)	2.123% (2.123%)	0.000%
	Current Users	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	HTTP 2D Image workflow	99.988% (99.988%)	0.000% (0.000%)	0.000% (0.000%)	0.012% (0.012%)	0.000%
	HTTP Semantic Dev workflow	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	HTTP Textual Search Dev workflow	97.811% (97.811%)	0.000% (0.000%)	0.000% (0.000%)	2.189% (2.189%)	0.000%
	HTTP Textual Search workflow	97.811% (97.811%)	0.000% (0.000%)	0.000% (0.000%)	2.189% (2.189%)	0.000%
	PING	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	Root Partition	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	SSH	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	Swap Usage	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	Total Processes	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
atos2-khresmoi	Current Load	96.959% (96.959%)	0.968% (0.968%)	0.000% (0.000%)	2.073% (2.073%)	0.000%
	Current Users	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	HTTP SCA Cooccurrence	99.977% (99.977%)	0.023% (0.023%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	HTTP SCA Dissambiguator	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	HTTP SCA KMI	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	HTTP SCA KMI Dev	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	HTTP SCA MT	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	HTTP SCA Mimir	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	HTTP SCA Mimir Dev	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	HTTP SCA ParaDISE	99.991% (99.991%)	0.000% (0.000%)	0.000% (0.000%)	0.009% (0.009%)	0.000%
	HTTP SCA Speller	99.873% (99.873%)	0.000% (0.000%)	0.000% (0.000%)	0.127% (0.127%)	0.000%
	PING	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	Root Partition	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	SSH	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	Swap Usage	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	Total Processes	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%

Figure 15: Nagios monthly availability report

As it can be seen above, the report not only reflects the status of the services and components developed for KHRESMOI in the selected VMs and nodes, but includes other not KHRESMOI-specific critical services that could potentially impact in the overall system performance. For instance, PING and SSH check the connectivity of the service, Current Load checks whether the load in terms of requests is below a predefined threshold, and so on and so forth. Therefore, this tool completes the toolset used to monitor the KHRESMOI Cloud, and helps to offer a wider vision of the system functionalities and real-time status.

7.4 Full Integrated infrastructure uses after Khresmoi

The Full Integrated Infrastructure has been defined and implemented to offer the required adaptability to new uses cases and environments. The SCA approach adopted allows gathering components and functionalities into different elements. These elements have been materialised in this project as Virtual Machines deployed on physical nodes, as shown in Figure 1. This implementation allows redeploying these services hosted on several VMs in different environments to fulfil new use cases. From the technical point of view, these services can be moved easily copying the *deployment.0* file associated to each VM. Working with different cloud hypervisor such as Xen[1] is enough to deploy these services in a new node. Besides the component services already hosted, the replication of these Virtual Machines preserves also all the software and network functionalities (like Java installation for instance).

Therefore, future production-like environments are feasible due to modularity and replicability features that current Full KHRESMOI Integrated infrastructure provides.

[1] http://www.xenproject.org/

8 Conclusion

This document aims to describe the Full KHRESMOI integrated infrastructure as the final iteration of the work developed during the project as well as showing a complete evaluation in order to check whether the platform fulfils the functional requirements collected. The main blocks of work implemented during the project are identified by the tasks and subtasks defined in the DoW[9] for Work Package 6, regarding architecture design, system scaling and system integration. Besides this, the evaluation has been conceived as a two part evaluation: firstly, assessing the scalability of the platform which represents all the work done regarding Full Cloud infrastructure, and secondly, testing integration of components, which englobes all work implemented in SCA component integration.

In order to do so, the evaluation reported follows the same approach of previous evaluations, providing a consistent vision over time of the assessment process. This assessment is based in a set of predefined metrics which have been used during the overall project evaluation and allow the comparison between different iterations and the evolution of the performance of the system.

These metrics have been used on the different KHRESMOI workflow scenarios covering the main uses cases since the first evaluations. The main outcomes after this last evaluation process are the following:

- The inherent complexity of the project integration has been solved successfully with SCA integration approach. Regarding functionality, SCA satisfies completely the requirements, but on downside of it is that performance is not completely satisfactory. This is due to the fact that the need of using several components and invocation calls among them affects negatively the performance, increasing Average Response Time in complex end-users requests.
- A Bigger set of functionalities has been included and evaluated. Most of these refinements have been specific requests from end-users, reviewers or partners (for instance the inclusion of Facets described in the document). In this sense, the Integration layer has shown great adaptability to include new functionalities into workflows composites.
- Similarly to integration solution, the scalability offered through the Full Cloud Prototype has been very useful during last year because it allows the reallocation of physical resources over the Virtual Machines. This feature allows re-assigning memory, CPU, or other resources to specific components depending on the demands detected through monitoring services.
- In this context, new monitoring tools and services have been included in response to petitions made by the reviewers of the project during the KHRESMOI 3rd technical review meeting. The main goal of these tools is checking availability of components and services, detecting patterns of platform use, detecting new requirements or updates on components, checking the platform traffic, etc. All this information represents an extra source of data that probes to be very helpful for the current and future scalability and adaptability of the system.

Summarizing, this document presented an exhaustive final evaluation for the Full KHRESMOI Integrated prototype. The prototype evaluated encompasses all the integrated work achieved during the KHRESMOI project, in order to create a platform capable to deal with initial project ambitions.

References

[1] Martinez Rodriguez, Ivan and Tinte Garcia, Miguel Angel, Deliverable D6.5.2 "Prototype and Evaluation of the 'Early integrated infrastructure'" in KHRESMOI Project, Seventh Framework Programme.

[2] Martinez Rodriguez, Ivan and Tinte Garcia, Miguel Angel, Deliverable 6.5.1 "Specification of the 'Early Integrated infrastructure'" in KHRESMOI Project, Seventh Framework Programme.

[3] Martinez Rodriguez, Ivan and Tinte Garcia, Miguel Angel, Deliverable D6.4.3 "Prototype and Evaluation of the 'Full Cloud Prototype'" in KHRESMOI Project, Seventh Framework Programme.

[4] Martinez Rodriguez, Ivan and Tinte Garcia, Miguel Angel, Deliverable D6.3.3 "Prototype and Evaluation of the 'Full Software Architecture'" in KHRESMOI Project, Seventh Framework Programme.

[5] Martinez Rodriguez, Ivan and Tinte Garcia, Miguel Angel, Deliverable D6.3.2 "Evaluation of the 'Early software architecture' and further specification" in KHRESMOI Project, Seventh Framework Programme.

[6] Martinez Rodriguez, Ivan and Tinte Garcia, Miguel Angel, Deliverable D6.4.2 "Evaluation of the "Early Cloud Infrastructure" and specification refinement for the "Full Cloud infrastructure"" in KHRESMOI Project, Seventh Framework Programme.

[7] Martinez Rodriguez, Ivan and Tinte Garcia, Miguel Angel, Deliverable D6.4.1 "State of the art, concepts and specification for the "Early Cloud infrastructure" in KHRESMOI Project, Seventh Framework Programme.

[8] Technical Review Report, KHRESMOI. Period No. 3, from 01-09-2012 to 31-08-2013. Authors: Dr. Pinar Wennerberg, Dr. Nicola Stokes, Mr. Pierre-Paul Sondag

[9] Description of Work, KHRESMOI. Seventh Programme Framework. THEME [ICT-2009.4.3] [Intelligent information management]

[10] T6.4.4 Refinement of the integrated and deployed infrastructure

[11] T6.4.5: Evaluation