



LarKC

The Large Knowledge Collider

a platform for large scale integrated reasoning and Web-search

FP7 – 215535

D11.1.2 LarKC metrics ontologies

Coordinator: Ioan Toma (Softgress)

With contributions from: Raluca Brehar (UTC), Silviu Bota (UTC), Mihai Negru (UTC), Andrei Vatavu (UTC), Mihai Chezan (Softgress)

Quality Assessor: Spyros Kotoulas (VUA)

Quality Controller: Sergiu Nedevschi (UTC)

Document Identifier:	LarKC/2008/D11.1.2/V1.0
Class Deliverable:	LarKC EU-IST-2008-215535
Version:	version 1.0.0
Date:	November 28, 2010
State:	final
Distribution:	public



EXECUTIVE SUMMARY

The goal of this document is to identify, to model and to formally represent as an ontology, the metrics that will be considered for instrumentation and monitoring of LarKC plugins, workflows and platform. For a system so complex as LarKC, instrumentation and monitoring can be done at different abstraction levels and can target different aspects of the platform and plugins. In this document, we identify for each of abstraction levels, sets of metrics and we describe each metric in detail in terms of what it represents, how it can be computed, its measurement unit, etc. A classification of these metrics is also provided. To give a precise semantic and formal definition of each metric, this document contains an ontology of metrics for LarKC instrumentation and monitoring that will allow for easy querying and report generation.



DOCUMENT INFORMATION

IST Project Number	FP7 – 215535	Acronym	LarKC
Full Title	The Large Knowledge Collider: a platform for large scale integrated reasoning and Web-search		
Project URL	http://www.larkc.eu/		
Document URL			
EU Project Officer	Stefano Bertolo		

Deliverable	Number	11.1.2	Title	LarKC metrics ontologies
Work Package	Number	11	Title	Instrumentation and Monitoring

Date of Delivery	Contractual	M32	Actual	30-Nov-10
Status	version 1.0.0		final <input checked="" type="checkbox"/>	
Nature	prototype <input type="checkbox"/> report <input checked="" type="checkbox"/> dissemination <input type="checkbox"/>			
Dissemination Level	public <input checked="" type="checkbox"/> consortium <input type="checkbox"/>			

Authors (Partner)	UTC, Softgress			
Resp. Author	Ioan Toma		E-mail	ioan.toma@softgress.com
	Partner	Softgress, UTC	Phone	+40 7676 84924

Abstract (for dissemination)	The goal of this document is to identify, to model and to formally represent as an ontology, the metrics that will be considered for instrumentation and monitoring of LarKC plugins, workflows and platform. For a system so complex as LarKC, instrumentation and monitoring can be done at different abstraction levels and can target different aspects of the platform and plugins. In this document, we identify for each of abstraction levels, sets of metrics and we describe each metric in detail in terms of what it represents, how it can be computed, its measurement unit, etc. A classification of these metrics is also provided. To give a precise semantic and formal definition of each metric, this document contains an ontology of metrics for LarKC instrumentation and monitoring that will allow for easy querying and report generation.
Keywords	Metrics, Measurement, Ontology, Instrumentation, Monitoring



PROJECT CONSORTIUM INFORMATION

Participant's name	Partner	Contact
Semantic Technology Institute Innsbruck, Universitaet Innsbruck	 	Prof. Dr. Dieter Fensel Semantic Technology Institute (STI), Universitaet Innsbruck, Innsbruck, Austria Email: dieter.fensel@sti-innsbruck.at
AstraZeneca AB		Bosse Andersson AstraZeneca Lund, Sweden Email: bo.h.andersson@astrazeneca.com
CEFRIEL - SOCIETA CONSORTILE A RESPONSABILITA LIMITATA		Emanuele Della Valle CEFRIEL - SOCIETA CONSORTILE A RE- SPONSABILITA LIMITATA Milano, Italy Email: emanuele.dellavalle@cefriel.it
CYCROP, RAZISKOVANJE IN EKSPERI- MENTALNI RAZVOJ D.O.O.		Michael Witbrock CYCROP, RAZISKOVANJE IN EKSPERI- MENTALNI RAZVOJ D.O.O., Ljubljana, Slovenia Email: witbrock@cyc.com
Höchstleistungsrechenzentrum, Universitaet Stuttgart		Georgina Gallizo Höchstleistungsrechenzentrum, Universitaet Stuttgart Stuttgart, Germany Email : gallizo@hlrs.de
MAX-PLANCK GESELLSCHAFT ZUR FOERDERUNG DER WISSENSCHAFTEN E.V.		Dr. Lael Schooler, Max-Planck-Institut für Bildungsforschung Berlin, Germany Email: schooler@mpib-berlin.mpg.de
Ontotext AD		Atanas Kiryakov, Ontotext Lab, Sofia, Bulgaria Email: naso@ontotext.com
SALTLUX INC.		Kono Kim SALTLUX INC Seoul, Korea Email: kono@saltlux.com
SIEMENS AKTIENGESELLSCHAFT		Dr. Volker Tresp SIEMENS AKTIENGESELLSCHAFT Muenchen, Germany Email: volker.tresp@siemens.com
THE UNIVERSITY OF SHEFFIELD		Prof. Dr. Hamish Cunningham, THE UNIVERSITY OF SHEFFIELD Sheffield, UK Email: h.cunningham@dcs.shef.ac.uk
VRIJE UNIVERSITEIT AMSTERDAM		Prof. Dr. Frank van Harmelen, VRIJE UNIVERSITEIT AMSTERDAM Amsterdam, Netherlands Email: Frank.van.Harmelen@cs.vu.nl
THE INTERNATIONAL WIC INSTI- TUTE, BEIJING UNIVERSITY OF TECHNOLOGY		Prof. Dr. Ning Zhong, THE INTERNATIONAL WIC INSTITUTE Mabeshi, Japan Email: zhong@maebashi-it.ac.jp
INTERNATIONAL AGENCY FOR RE- SEARCH ON CANCER		Dr. Paul Brennan, INTERNATIONAL AGENCY FOR RE- SEARCH ON CANCER Lyon, France Email: brennan@iarc.fr
INFORMATION RETRIEVAL FACILITY		Dr. John Tait, Dr. Paul Brennan, INFORMATION RETRIEVAL FACILITY Vienna, Austria Email: john.tait@ir-facility.org





TECHNICAL UNIVERSITY OF CLUJ-NAPOCA http://www.utcluj.ro/	 The logo of the Technical University of Cluj-Napoca, consisting of a stylized 'U' and 'T' in red and grey.	Prof. Dr. Eng. Sergiu Nedevschi TECHNICAL UNIVERSITY OF CLUJ-NAPOCA Cluj-Napoca, Romania E-mail: sergiu.nedevschi@cs.utcluj.ro
SOFTGRESS S.R.L. http://www.softgress.com/	 The logo for Softgress, featuring the word 'Softgress' in white text on a blue rectangular background.	Dr. Ioan Toma SOFTGRESS S.R.L. Cluj-Napoca, Romania E-mail: ioan.toma@softgress.com



TABLE OF CONTENTS

LIST OF FIGURES	8
1 INTRODUCTION	9
2 INSTRUMENTATION LEVELS	10
2.1 General description of the metrics	10
2.1.1 Performance	10
2.1.2 Resources	11
2.1.3 Data	11
2.1.4 Invocations / Instances	11
2.1.5 Events	11
2.2 Platform instrumentation and monitoring	11
2.2.1 Queries	12
2.2.2 Workflows	12
2.2.3 Platform components	12
2.3 Plugin instrumentation and monitoring	12
2.4 Instrumentation context	13
3 METRICS CLASSIFICATION	14
3.1 Atomic metrics	14
3.2 Compound metrics	15
4 METRICS DESCRIPTIONS	16
4.1 Atomic metrics	16
4.1.1 Query related metrics	16
4.1.2 Workflow metrics	19
4.1.3 Plugin related metrics	19
4.1.4 Platform related metrics	22
4.2 Compound metrics	22
4.2.1 Query related metrics	23
4.2.2 Workflow related metrics	24
4.2.3 Plugin related metrics - Method Invocation	25
4.2.4 Plugin related metrics - I/O RDF Triples	25
4.2.5 Plugin related metrics - Execution Time	26
4.2.6 Plugin related metrics - Threads	27
4.2.7 Plugin related metrics - Nodes	28
4.2.8 Plugin related metrics - Exceptions	29
4.2.9 Plugin related metrics - Data Layer Accesses	29
4.2.10 Plugin related metrics - Cache Hit and Miss	31
4.2.11 Platform compound metrics	33
4.3 Generic metrics	34
4.3.1 Method related metrics	34
4.3.2 JVM related metrics	35
4.3.3 System related metrics	36
4.3.4 Instrumented application related metrics	39
5 ONTOLOGY OF METRICS	41



6 CONCLUSION

59



LIST OF FIGURES

2.1	Platform instrumentation level	12
2.2	plugin instrumentation level	13



1. Introduction

The aim of the LarKC project is to develop a distributed platform for incomplete reasoning that will scale far beyond the level of current reasoning systems. LarKC has a plugable architecture allowing users to plug their own components, known as plugins in LarKC terminology, into the platform, to run and to test them by performing large scale experiments. As with any distributed system, a vertical service that is always required is instrumentation and monitoring of the system components and of the system as a whole. Such a service enables us to observe if, and how well the system and its components are performing. In particular for LarKC, developers would like to know for their plugins and components how performant they are, how many resources they consume, how many times these plugins and components have been invoked and used, how much data they consume and produce, etc. Answers for all these questions, can be provided by what is known as *instrumentation and monitoring* solutions.

In a nutshell, instrumentation is the process of introducing new code in key parts of the application that once executed performs measurements which then are made available to the monitoring system. In other words, instrumentation provides code that enables measurements of various metrics that are of interest for user. Monitoring, on the other hand is the process of performing the measurements and making the user of the system aware of the measurements.

The LarKC platform developed so far provides limited support for instrumentation and monitoring and thus is rather difficult for LarKC users to evaluate how well their plugins are performing. The lack of instrumentation and monitoring at the level of the platform, makes difficult for LarKC platform developers as well to assess how well the LarKC platform itself is performing. In WP11, we address these limitations by building an instrumentation and monitoring solution for LarKC that will be integrated with the LarKC platform.

The goal of this document is to identify, to model and to formally represent as an ontology, the metrics that will be considered for instrumentation and monitoring of LarKC plugins, workflows and platform. For a system so complex as LarKC, instrumentation and monitoring can be done at different abstraction levels and can target different aspects of the platform and plugins. In this deliverable we identify, for each of the abstraction levels, sets of metrics and we describe each metric in details in terms of what it represents, how it can be computed, its measurement unit, etc. A classification of these metrics is also provided. To give a precise semantic and formal definition of each metric, this deliverable contains as an ontology of metrics for LarKC instrumentation and monitoring. The metrics and relations between them are represented as concepts and relations in an ontology. This will allow for easy querying and report generation of monitored data.

This deliverable is organized as follows. Chapter 2 identifies and provides models for instrumentation and monitoring metrics that are relevant for both plugin and platform. The identified metrics can be classified according to various criteria such as complexity (i.e. is the metric simple and directly measured by the instrumentation or is it obtained by combining other metrics) and specificity (i.e. is the metric generic for any application or is it specific to LarKC). This discussion is performed in Chapter 3. Chapter 4 contains a detailed definition of each of the metrics, while Chapter 5 contains the formalization of the instrumentation and monitoring metrics introduced in Chapter 4 as an OWL ontology. Finally, Chapter 6 concludes the deliverable.



2. Instrumentation Levels

In this chapter, we identify and provide models for instrumentation and monitoring metrics that are relevant for both plugins and platform. The metrics are identified based on the requirements defined in [1].

The LarKC platform was conceived to be a flexible plugin architecture which enables design and testing of new reasoning techniques. Reasoning over large knowledge bases is ensured by the plugin composition mechanism. Several types of plugins are grouped in a workflow that solves different reasoning tasks. The plugins arrangement in a workflow requires some support for composition, instantiation, monitoring and control, which is provided by the platform.

The instrumentation and monitoring is closely related to the design and architecture of the LarKC platform and plugins, hence we distinguish between two major levels of instrumentation i.e. plugin and platform. For each instrumentation level we associate a set of metrics. This chapter will describe informally each category of metrics and it will show what are the components of the current architecture that can provide instrumentation information. The formal structure of our metrics model will be defined in Chapter 3 and a clear definition of each proposed metric will be presented by Chapter 4. Chapter 5 will further show how the proposed metrics can be modeled as part of an ontology and it will provide the RDF/S representation of LarKC metrics ontology.

2.1 General description of the metrics

Informally the metrics that will be gathered by the instrumentation process can be grouped into several categories:

- Performance
- Resources
- Invocations / Instances
- Data
- Events

2.1.1 Performance

The metrics in this category are related to how fast a request/task can be completed and at which rate the results are generated i.e. we may include in this category:

- Execution time: measures the time during which actual work has been carried out.
- Throughput: estimated rate of generated results.

Performance can be measured from different perspectives and therefore, a number of different parameters will be included in this group.



2.1.2 Resources

The metrics in this group are related to the usage of resources and the infrastructure where the plugins are executed. This category may comprise the following:

- Memory consumption: the memory required for the execution of a plugin/workflow.
- CPU usage
- Number of threads
- Number of nodes in a grid

2.1.3 Data

The metrics in this category refer to the size of the data that can be input or output for a task/method or transmitted between plugins. The measurement units could be bytes or number of triples contained.

2.1.4 Invocations / Instances

The metrics in this category are quantity metrics, because the information that they contain refers to counting invocations of a method, plugin, workflow or instances of something. We may count the number of instantiations of a given plugin, the number of queries that were given to the platform in a certain amount of time, the number of accesses to the data layer, how many successful executions have been run, how many failures etc.

2.1.5 Events

Special exceptions and errors will also be recorded in a category of metrics. The events may mark a successful execution, a failure, a thrown exception.

2.2 Platform instrumentation and monitoring

The most general level on which instrumentation and monitoring can be performed is at the platform level. All the measurements that are done at this point refer to the platform and the functionalities it offers, that is plugin composition, instantiation, data transfer and communication between plugins and the data layer. We consider that the current design and architecture of LarKC comprises several instrumentation information providers that can be:

- queries
- workflows
- platform components

A schematic figure of what instrumentation information can be gathered at platform level is given in Figure 2.1.

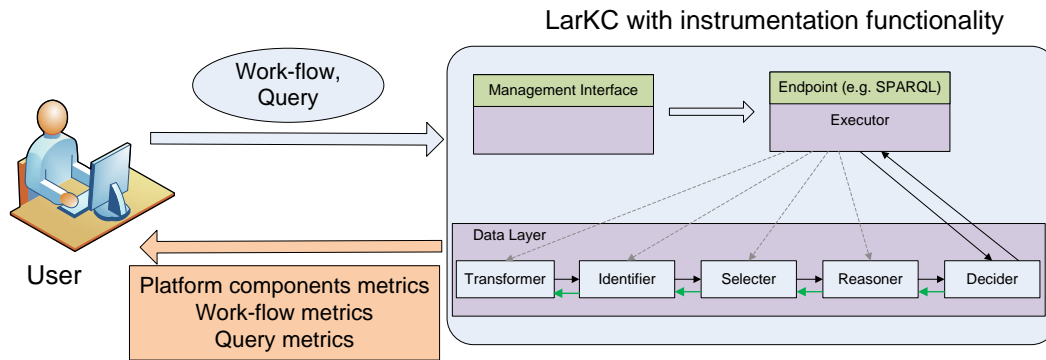


Figure 2.1: Platform instrumentation level

2.2.1 Queries

The instrumentation information in a query can be provided by SPARQL analysis. Hence, the query may tell us how many name-spaces are used, how many variables, how many RDF triples. Although this data may seem redundant we intend to use it in the experiments we plan with the relevance feedback mechanism (see the scenarios in [1]).

2.2.2 Workflows

For the workflow we may measure the total execution time, we may provide a summary of the total or average resources that have been used, give information about the data transmission within the workflow, record the exceptions or failures, count how many types of plugins have been run in a given workflow.

2.2.3 Platform components

The metrics applicable to the whole platform are very general and provide overall information about the execution time of the platform, the number of workflows that have been run during the execution of the platform, the number of queries that have been provided to the platform, count how many instances of different plugin types have been run successfully or failed.

2.3 Plugin instrumentation and monitoring

A more specific or granular level on which instrumentation and monitoring can be done is the plugin level. All the measurements that are performed at this point refer to the behavior of a plugin in different contexts. Figure 2.2 depicts a schematic view of the plugin instrumentation level.

For each plugin we may record information about its type (i.e. decider, reasoner, selector, identifier, transformer), the performance parameters, the resources it has used, the input / output data, the number of accesses to the data layer or the failure of the execution.

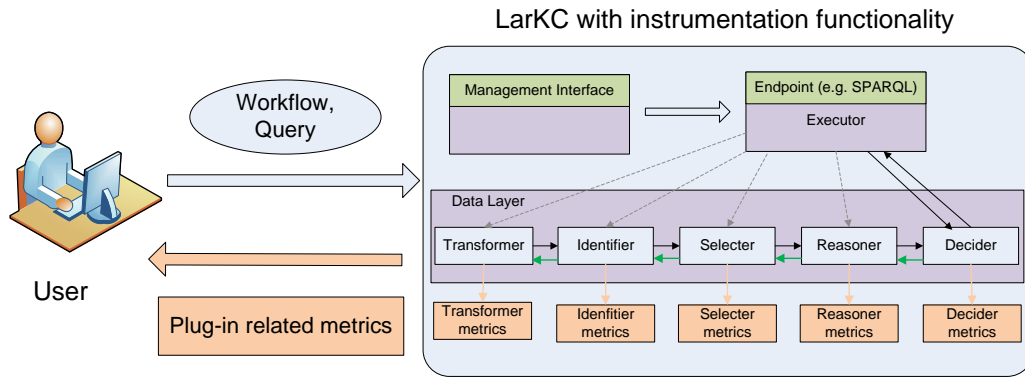


Figure 2.2: plugin instrumentation level

2.4 Instrumentation context

The context captures information about the environment in which the measurement is done. The instrumentation context represents important information that will guide our decisions for the design and architecture of the instrumentation solution. On each instrumentation level we may define specific contexts. For example the context of a plugin in a workflow may be given by the workflow name and a unique query identifier. The context of a workflow in a platform may be given by the plugins that form the workflow, by the available resources at the time the query is given to it, by the size of the query, etc.



3. Metrics Classification

In this chapter, we briefly introduce different classes of metrics. We have identified two classes of metrics: atomic and compound metrics. Atomic metrics are measured and recorded during the execution of the LarKC Platform. Compound metrics are computed off-line, outside the LarKC Platform and are recorded together with the atomic metrics in our metric database. They usually consist in an aggregation of one or more atomic metrics, but they could also be an aggregation of other compound metrics. In the next sections of this chapter we will enumerate and explain the attributes of each class of metrics.

3.1 Atomic metrics

An atomic metric is a metric that can be directly measured inside the LarKC Platform. The measurement and recording of such a metric should not affect the overall performance of the LarKC Platform. An atomic metric is characterized by:

- metric identifier
- metric type
- metric time stamp
- a list of tags (context in which the measurement was done)
- metric value
- metric measurement unit (not all the metrics have a measurement unit).

The metric identifier is a string that identifies the measurement that will be carried out.

We have identified several atomic metric types:

1. **EVENT** - Some code point has been reached and the time stamp at which this event occurred is transmitted. An event metric has no value.
2. **DURATION** - The time between two events (e.g. the time it took for some code to be executed).
3. **CPU_LOAD** - The load of the CPU at the time of measurement (per computation node).
4. **MEMORY_PERCENT** - The memory load at the time of the measurement (per computation node).
5. **DATA_SIZE** - The size of the data structure (RDF triples set that are transmitted between plugins, etc.); measured in bytes.
6. **NUMBER_OF_INSTANCES** - The number of some object instances (inside LarKC number of plugins, threads, number of RDF triples, number of queries per workflow, workflows per platform, etc.).



7. FAILURE_SUCCESS - The execution was terminated successfully or with a failure (may be included as an event's subcategory).
8. EXCEPTION - An exceptional condition occurred while processing the request (may be included as an event's subcategory).
9. VALUE - The value of some variable in LarKC (predefined values, can not be set by the user).

Every atomic metric is identified by a list of tags. The same is true for compound metrics. This list of tags identifies the context of the measurement.

1. *Query* – identified by the TCP/IP session established when a query is made to the LarKC platform (request, response). It represents an unique identifier.
2. *Workflow* – identified by the workflow name or type / identified by the list of plugins involved in solving a given query (number and type of plugins).
3. *Plugin* – identified by plugin name.
4. *Method* – identifies the method invoked / method name.

The metric time stamp represents the time when the measurement is being recorded; the metric value represents the value of the measured atomic metric.

We have identified several metric measurement units: micro-seconds, bytes, percent, RDF triple.

3.2 Compound metrics

Any compound metric (that can be computed by aggregation of several atomic metrics) is characterized by:

- metric identifier
- metric type
- one or more tags, or no tags
- metric value
- metric measurement unit

The metric identifier, measurement unit and value have the same meaning as in the case of the atomic metrics. The compound metric can contain one or more tags from the atomic metrics that it aggregates, or it can contain no tags meaning that it can be a platform related metric.

The compound metric types are presented below:

1. NbEventsPerTimeUnit = Number of events divided by the difference between the last and first time stamp.
2. AvgDimension - average dimension of a number of instances (average memory usage, average CPU usage, average time) = Sum of dimensions divided by the number of instances during a time interval.
3. TotalDimension - total dimension of a number of instances (total memory usage, total time) = Sum of dimension during a time interval.



4. Metrics Descriptions

As discussed in Chapter 3 there are basically two criteria based on which we can classify various metrics the complexity and the specificity of a metric. The first criteria, namely complexity is about how metric measurements are obtained. If the metric is directly measured by the instrumentation code we call such metrics, simple metrics or *atomic metrics*. If the metric is obtained by combining other metrics we call such metrics *compound metrics*. The second criteria, namely specificity, refers to how general or specific a metric is with respect to the system being instrumented, LarKC in our case. As mentioned in Chapter 3, we distinguish between two categories of metrics i.e. *generic metrics* that are common to many java applications and *LarKC specific metrics* that are specific to LarKC plugins, workflow and platform. Please note that the metrics belonging to one category according to one criteria can be part of another category according to the other criteria. For example an atomic metric can be as well a LarKC specific metric.

In the rest of this chapter we describe each of the metrics identified in Chapter 3 from a conceptual point of view. Each metric is described in details, including information such as the type of the metric, the tags that can be associated with the metrics and the measurement unit of the metric. The metrics will be introduced and grouped based on the classification used in Chapter 3.

4.1 Atomic metrics

4.1.1 Query related metrics

The atomic metrics in this category capture the information that can be extracted from a query or from the answer to the query. All the metrics have the Query tag set. We will consider the following:

1. *QueryInvocation*: records the fact that a new query has been received by the platform.
 - Type: EVENT
 - Tags: Query, Workflow, Platform
 - Measurement unit: –
2. *QueryContent*: stores the query that was received by the LarKC platform.
 - Type: VALUE (string)
 - Tags: Query, Workflow, Platform
 - Measurement unit: –
3. *QuerySizeInBytes*: represents the size of the query in bytes.
 - Type: DATA_SIZE (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: bytes
4. *QuerySizeInTriples*: denotes the number of RDF triples that are contained in the query.



- Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: RDF triples.
5. *QueryNamespacesNb*: stands for the number of PREFIX statements that are given in the query.
- Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: –
6. *QueryNamespace*: the value of this metric contains the name-spaces used in the query.
- Type: VALUE (string)
 - Tags: Query, Workflow, Platform
 - Measurement unit: –
7. *QueryVariablesNb*: counts the number of variables in the query.
- Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: –
8. *QueryDataSetSourcesNb*: the amount of FROM clauses in the query.
- Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: –
9. *QueryOperatorsNb*: represents the number of logical, arithmetic and RDF operators.
- Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: –
10. *QueryResultOrderingNb*: expresses the number of fields in the ORDER BY statement.
- Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: –
11. *QueryResultLimitNb*: gives the value of n from the SPARQL statement LIMIT n .
- Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Platform



- Measurement unit: –
12. *QueryResultOffsetNb*: gives the value of m from the SPARQL statement OFFSET m .
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: –
 13. *QueryResultInTriples*: stands for the amount of RDF triples contained in the result.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: RDF triples
 14. *QueryResultInBytes*: symbolizes the size in bytes of the result.
 - Type: DATA_SIZE (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: bytes
 15. *QueryTotalResponseTime*: denotes the time elapsed from when the query was sent to the platform until the final result is provided to the user.
 - Type: DURATION (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: microseconds
 16. *QueryCompletionStatus*: indicates if the query has ended with success or the execution provided an exception (failure). If the execution of a query gives an exception then the query completion status is a failure, otherwise it is a success.
 - Type: FAILURE_SUCCESS (boolean)
 - Tags: Query, Workflow, Platform
 - Measurement unit: –
 17. *QueryBlockedTime*: denotes the Amount of time spent in blocked state while executing query.
 - Type: DURATION (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: microseconds



4.1.2 Workflow metrics

The atomic metrics comprised in this class provide measurement information that is related to a given workflow.

1. *WorkflowInvocation*: A new workflow is started by the LarKC platform.
 - Type: EVENT
 - Tags: Query, Workflow, Platform
 - Measurement unit: –
2. *WorkflowThreadNb*: symbolizes the total number of threads started during the execution of the workflow.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: –
3. *WorkflowPluginsNb*: indicates the total number of plugins that were involved in the execution of the workflow.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: –
4. *WorkflowPluginTypeNb*: the number of identical plugins that are involved in an execution of the workflow.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: –
5. *WorkflowDuration*: indicates the total execution time of the workflow.
 - Type: DURATION (integer)
 - Tags: Query, Workflow, Platform
 - Measurement unit: microseconds

4.1.3 Plugin related metrics

The atomic metrics comprised in this category offer measurement information that is related to a given plugin.

1. *PluginInputSizeInTriples*: denotes the number of RDF statements that are input to the plugin.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Plugin, Platform
 - Measurement unit: RDF triples



2. *PluginOutputSizeInTriples*: represents the number of RDF statements that are output by the plugin.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Plugin, Platform
 - Measurement unit: RDF triples
3. *PluginInputSizeInBytes*: symbolizes the size in bytes of the RDF Triples that are input to the plugin.
 - Type: DATA_SIZE (integer)
 - Tags: Query, Workflow, Plugin, Platform
 - Measurement unit: bytes
4. *PluginOutputSizeInBytes*: indicates the size in bytes of the RDF Triples that are output by the plugin.
 - Type: DATA_SIZE (integer)
 - Tags: Query, Workflow, Plugin, Platform
 - Measurement unit: bytes
5. *PluginTotalExecutionTime*: denotes the total execution time of a plugin inside the workflow.
 - Type: DURATION (integer)
 - Tags: Query, Workflow, Plugin, Platform
 - Measurement unit: microseconds
6. *PluginThreadsStartedNb*: the number of threads started by the plugin.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Plugin, Platform
 - Measurement unit: –
7. *PluginNodesNb*: the number of nodes that the plugin is executed on.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: Query, Workflow, Plugin, Platform
 - Measurement unit: –
8. *PluginMethodInvocation*: a method inside a plugin is called.
 - Type: EVENT
 - Tags: Query, Workflow, Plugin, Method, Platform
 - Measurement unit: –
9. *PluginException*: marks an uncaught exception in a plugin.
 - Type: EXCEPTION (string)
 - Tags: Query, Workflow, Plugin, Platform



- Measurement unit: –
10. *PluginDataLayerAccess*: the size in bytes of the returned data after an access to the data layer, during the execution of the plugin.
 - Type: DATA_SIZE (integer)
 - Tags: Query, Workflow, Plugin, Platform
 - Measurement unit: bytes
 11. *PluginInstanceCreation*: the instantiation of a plugin by the Executer.
 - Type: EVENT
 - Tags: Query, Workflow, Plugin, Platform
 - Measurement unit: –
 12. *PluginBlockedTime*: the amount of time spent in blocked state while executing the plugin.
 - Type: DURATION
 - Tags: Query, Workflow, Plugin, Platform
 - Measurement unit: microseconds
 13. *PluginGarbageCollectionTime*: the amount of time spent collecting the garbage in this plugin.
 - Type: DURATION
 - Tags: Query, Workflow, Plugin, Platform
 - Measurement unit: microseconds
 14. *PluginCacheHit*: a cache hit was produced in a plugin when accessing the data layer.
 - Type: EVENT
 - Tags: Query, Workflow, Plugin, Platform
 - Measurement unit: –
 15. *PluginCacheMiss*: a cache miss was produced in a plugin when accessing the data layer.
 - Type: EVENT
 - Tags: Query, Workflow, Plugin, Platform
 - Measurement unit: –



4.1.4 Platform related metrics

The atomic metrics comprised in this group supply measurement information that is related to the platform.

1. *PlatformExecutionTime*: the time elapsed since the platform was started until its execution has ended.
 - Type: DURATION (long)
 - Tags: Platform
 - Measurement unit: microseconds
2. *PlatformCPULoad*: the CPU LOAD of the LarKC platform at a given time-stamp.
 - Type: CPU_LOAD (float)
 - Tags: Platform
 - Measurement unit: percent
3. *PlatformMemoryUsage*: the memory percent usage of the platform at a given time-stamp.
 - Type: MEMORY_PERCENT (float)
 - Tags: Platform
 - Measurement unit: percent
4. *PlatformMemoryDim*: the amount of memory used by the platform at a given time-stamp.
 - Type: DATA_SIZE (integer)
 - Tags: Platform
 - Measurement unit: bytes
5. *PlatformGarbageCollectionTime*: the amount of time spent for the whole platform while collecting garbage
 - Type: DURATION (integer)
 - Tags: Platform
 - Measurement unit: microseconds

4.2 Compound metrics

This section presents a list of compound metrics, their description and the method of obtaining these metrics.



4.2.1 Query related metrics

1. *QueriesPerTimeIntervalNb*: counts the number of queries that were received by the LarKC platform in a given time interval.
 - Required Input: Time Interval (T)
 - Type: NbEventsPerTimeUnit (integer)
 - Tags: Query
 - Measurement Unit: –
 - Formula: $Count_T(QueryInvocation)$
2. *QueriesPerNamespacePerTimeIntervalNb*: denotes the number of queries that contained a given name space (PREFIX) and were received by the LarKC platform in a time interval.
 - Required Input: Name Space (NS), Time Interval (T)
 - Type: NbEventsPerTimeUnit (integer)
 - Tags: Query
 - Measurement Unit: –
 - Formula: $Count_{NS,T}(QueryInvocation)$
3. *QueriesWithExecutionTimeInRangePerTimeIntervalNb*: represents the number of queries that have the response time in a given range of limits.
 - Required Input: Execution Time Range (ETR), Time Interval (T)
 - Type: NbEventsPerTimeUnit (integer)
 - Tags: Query
 - Measurement Unit: –
 - Formula: $Count_{ETR,T}(QueryTotalResponseTime)$
4. *QuerySuccessRatePerTimeInterval*: indicates the amount of queries that finished with success. At the level of instrumentation and monitoring it is hard to figure out if the results returned by the query are of good quality. We consider thus a query to be successful if there were no exceptions during the execution of this query.
 - Required Input: Time Interval (T)
 - Type: NbEventsPerTimeUnit (integer)
 - Tags: Query
 - Measurement Unit: –
 - Formula: $Count_T(QueryCompletionStatus = SUCCESS)$
5. *QueryFailureRatePerTimeInterval*: symbolizes the number of queries that finished with failure.
 - Required Input: Time Interval (T)
 - Type: NbEventsPerTimeUnit (integer)



- Tags: Query
- Measurement Unit: –
- Formula: $Count_T(QueryCompletionStatus = FAILURE)$

4.2.2 Workflow related metrics

1. *WorkflowsPerTimeIntervalNb*: represents the number of workflows of a given name that were started in a given time interval.
 - Required Input: Workflow name (W), Time Interval (T)
 - Type: NbEventsPerTimeUnit (integer)
 - Tags: Workflow
 - Measurement Unit: –
 - Formula: $Count_{W,T}(WorkflowInvocation)$
2. *WorkflowAvgNodesPerTimeIntervalNb*: denotes the average number of nodes used by a given workflow in a given time interval.
 - Required Input: Workflow name (W), Time Interval (T)
 - Type: AvgDimension (float)
 - Tags: Workflow
 - Measurement Unit: –
 - Formula: $\frac{\sum_{W,T}(WorkflowGridNodesNb)}{WorkflowsPerTimeIntervalNb}$
3. *WorkflowAvgThreadsPerTimeIntervalNb*: The average number of threads used by a given workflow in a given time interval.
 - Required Input: Workflow name (W), Time Interval (T)
 - Type: AvgDimension (float)
 - Tags: Workflow
 - Measurement Unit: –
 - Formula: $\frac{\sum_{W,T}(WorkflowThreadNb)}{WorkflowsPerTimeIntervalNb}$
4. *WorkflowAvgDurationPerTimeInterval*: indicates the average duration of a given workflow in a given time interval.
 - Required Input: Workflow name (W), Time Interval (T)
 - Type: AvgDimension (float)
 - Tags: Workflow
 - Measurement Unit: microseconds
 - Formula: $\frac{\sum_{W,T}(WorkflowDuration)}{WorkflowsPerTimeIntervalNb}$



4.2.3 Plugin related metrics - Method Invocation

1. *PluginMethodInvocationPerTimeIntervalNb*: counts the number of invocations of a given method name in a given plugin in a given workflow.
 - Required Input: Workflow name (W), Plugin name (P), Method name (M), Time Interval (T)
 - Type: NbEventsPerTimeUnit (integer)
 - Tags: Workflow, Plugin, Method
 - Measurement Unit: –
 - Formula: $Count_{W,P,M,T}(PluginMethodInvocation)$
2. *PluginInstanceCreationPerTimeIntervalNb*: represents how many instances of a given plugin were created in a given time interval.
 - Required Input: Plugin name (P), Time Interval (T)
 - Type: NbEventsPerTimeUnit (integer)
 - Tags: Plugin
 - Measurement Unit: –
 - Formula: $Count_{P,T}(PluginInstanceCreation)$

4.2.4 Plugin related metrics - I/O RDF Triples

1. *PluginAvgInputSizeInTriplesPerTimeInterval*: indicates the average number of RDF Triples that are provided as input to a specified plugin in a given time interval.
 - Required Input: Plugin name (P), Time Interval (T)
 - Type: AvgDimension (float)
 - Tags: Plugin
 - Measurement Unit: –
 - Formula: $\frac{\sum_{P,T}(PluginInputSizeInTriples)}{PluginInstanceCreationPerTimeIntervalNb}$
2. *PluginAvgOutputSizeInTriplesPerTimeInterval*: denotes the average number of RDF Triples that are provided as output by specified plugin in a given time interval.
 - Required Input: Plugin name (P), Time Interval (T)
 - Type: AvgDimension (float)
 - Tags: Plugin
 - Measurement Unit: –
 - Formula: $\frac{\sum_{P,T}(PluginOutputSizeInTriples)}{PluginInstanceCreationPerTimeIntervalNb}$
3. *PluginAvgInputSizeInBytesPerTimeInterval*: The average memory dimension of the RDF Triples that are provided as input to a specified plugin in a given time interval.



- Required Input: Plugin name (P), Time Interval (T)
 - Type: AvgDimension (float)
 - Tags: Plugin
 - Measurement Unit: bytes
 - Formula:
$$\frac{\sum_{P,T}(PluginInputSizeInBytes)}{PluginInstanceCreationPerTimeIntervalNb}$$
4. *PluginAvgOutputSizeInBytesPerTimeInterval*: The average memory dimension of the RDF Triples that are provided as output by the given Decider plugin in a given time interval.
- Required Input: Plugin name (P), Time Interval (T)
 - Type: AvgDimension (float)
 - Tags: Plugin
 - Measurement Unit: bytes
 - Formula:
$$\frac{\sum_{P,T}(PluginOutputSizeInBytes)}{PluginInstanceCreationPerTimeIntervalNb}$$

The above compound metrics are measured at platform level. We can derive similar measures at workflow level, for several runs of a given workflow during a time interval.

4.2.5 Plugin related metrics - Execution Time

1. *WorkflowPluginTotalExecutionTimePerTimeInterval*: Total execution time of the given plugin in all the runs of a specified workflow in a given time interval.
- Required Input: Workflow name (W), Plugin name (P), Time Interval (T)
 - Type: TotalDimension (integer)
 - Tags: Workflow, Plugin
 - Measurement Unit: microseconds
 - Formula:
$$\sum_{W,P,T}(PluginTotalExecutionTime)$$
2. *PlatformPluginTotalExecutionTimePerTimeInterval*: Total execution time of the given plugin in a given time interval, in all the workflows.
- Required Input: Plugin name (P), Time Interval (T)
 - Type: TotalDimension (integer)
 - Tags: Plugin
 - Measurement Unit: microseconds
 - Formula:
$$\sum_{P,T}(PluginTotalExecutionTime)$$
3. *WorkflowPluginAvgExecutionTimePerTimeInterval*: Average execution time of the given plugin in all the runs of a specified workflow in a given time interval.
- Required Input: Workflow name (W), Plugin name (P), Time Interval (T)
 - Type: AvgDimension (float)
 - Tags: Workflow, Plugin



- Measurement Unit: microseconds
 - Formula: $\frac{WorkflowPluginTotalExecutionTimePerTimeInterval}{WorkflowsPerTimeIntervalNb * WorkflowPluginTypeNb}$
4. *PlatformPluginAvgExecutionTimePerTimeInterval*: Average execution time of the given plugin in a given time interval, in all the workflows.
- Required Input: Plugin name (P), Time Interval (T)
 - Type: AvgDimension (float)
 - Tags: Plugin
 - Measurement Unit: microseconds
 - Formula: $\frac{PlatformPluginTotalExecutionTimePerTimeInterval}{PluginInstanceCreationPerTimeIntervalNb}$

4.2.6 Plugin related metrics - Threads

1. *WorkflowPluginTotalThreadsStartedPerTimeIntervalNb*: Total number of threads started by the given plugin in all the runs of a specified workflow in a given time interval.
- Required Input: Workflow name (W), Plugin name (P), Time Interval (T)
 - Type: TotalDimension (integer)
 - Tags: Workflow, Plugin
 - Measurement Unit: –
 - Formula: $\sum_{W,P,T}(PluginThreadsStartedNb)$
2. *PlatformPluginTotalThreadsStartedPerTimeIntervalNb*: Total number of threads started by the given plugin in a given time interval, in all the workflows.
- Required Input: Plugin name (P), Time Interval (T)
 - Type: TotalDimension (integer)
 - Tags: Plugin
 - Measurement Unit: –
 - Formula: $\sum_{P,T}(PluginThreadsStartedNb)$
3. *WorkflowPluginAvgThreadsStartedPerTimeIntervalNb*: Average number of threads started by the given plugin in all the runs of a specified workflow in a given time interval.
- Required Input: Workflow name(W), Plugin name (P), Time Interval (T)
 - Type: AvgDimension (float)
 - Tags: Workflow, Plugin
 - Measurement Unit: –
 - Formula: $\frac{WorkflowPluginTotalThreadsStartedPerTimeIntervalNb}{WorkflowsPerTimeIntervalNb * WorkflowPluginTypeNb}$
4. *PlatformPluginAvgThreadsStartedPerTimeIntervalNb*: Average number of threads started by the given plugin in a given time interval, in all the workflows.
- Required Input: Plugin name (P), Time Interval (T)



- Type: AvgDimension (float)
- Tags: Plugin
- Measurement Unit: –
- Formula: $\frac{PlatformPluginTotalThreadsStartedPerTimeIntervalNb}{PluginInstanceCreationPerTimeIntervalNb}$

4.2.7 Plugin related metrics - Nodes

1. *WorkflowPluginTotalNodesPerTimeIntervalNb*: Total number of nodes used by the given plugin in all the runs of a specified workflow in a given time interval.
 - Required Input: Workflow name (W), Plugin name (P), Time Interval (T)
 - Type: TotalDimension (integer)
 - Tags: Workflow, Plugin
 - Measurement Unit: –
 - Formula: $\sum_{W,P,T}(PluginNodesNb)$
2. *PlatformPluginTotalNodesPerTimeIntervalNb*: Total number of nodes used by the given plugin in a given time interval, in all the workflows.
 - Required Input: Plugin name (P), Time Interval (T)
 - Type: TotalDimension (integer)
 - Tags: Plugin
 - Measurement Unit: –
 - Formula: $\sum_{P,T}(PluginNodesNb)$
3. *WorkflowPluginAvgNodesPerTimeIntervalNb*: Average number of nodes used by the given plugin in all the runs of a specified workflow in a given time interval.
 - Required Input: Workflow name(W), Plugin name (P), Time Interval (T)
 - Type: AvgDimension (float)
 - Tags: Workflow, Plugin
 - Measurement Unit: –
 - Formula: $\frac{WorkflowPluginTotalNodesPerTimeIntervalNb}{WorkflowsPerTimeIntervalNb*WorkflowPluginTypeNb}$
4. *PlatformPluginAvgNodesPerTimeIntervalNb*: Average number of nodes used by the given plugin in a given time interval, in all the workflows.
 - Required Input: Plugin name (P), Time Interval (T)
 - Type: AvgDimension (float)
 - Tags: Plugin
 - Measurement Unit: –
 - Formula: $\frac{PlatformPluginTotalNodesPerTimeIntervalNb}{PluginInstanceCreationPerTimeIntervalNb}$



4.2.8 Plugin related metrics - Exceptions

1. *WorkflowPluginExceptionPerTimeIntervalNb*: Total number of exceptions recorded by a given plugin in all the runs of a given workflow in a time interval.
 - Required Input: Workflow name (W), Plugin name (P), Time Interval (T)
 - Type: TotalDimension (integer)
 - Tags: Workflow, Plugin
 - Measurement Unit: –
 - Formula: $Count_{W,P,T}(PluginException)$
2. *PlatformPluginExceptionPerTimeIntervalNb*: Total number of exceptions recorded by a given plugin in a time interval, in all the workflows.
 - Required Input: Plugin name (P), Time Interval (T)
 - Type: TotalDimension (integer)
 - Tags: Plugin
 - Measurement Unit: –
 - Formula: $Count_{P,T}(PluginException)$

4.2.9 Plugin related metrics - Data Layer Accesses

1. *WorkflowPluginTotalDataLayerAccessPerTimeIntervalNb*: The total number of data layer accesses from a given plugin for a given workflow in a given time interval.
 - Required Input: Workflow name (W), Plugin name (P), Time Interval (T)
 - Type: NbEventsPerTimeUnit (integer)
 - Tags: Workflow, Plugin
 - Measurement Unit: –
 - Formula: $Count_{W,P,T}(PluginDataLayerAccess)$
2. *PlatformPluginTotalDataLayerAccessPerTimeIntervalNb*: The total number of data layer accesses from a given plugin in a given time interval, in all the workflows.
 - Required Input: Plugin name (P), Time Interval (T)
 - Type: NbEventsPerTimeUnit (integer)
 - Tags: Plugin
 - Measurement Unit: –
 - Formula: $Count_{P,T}(PluginDataLayerAccess)$
3. *WorkflowPluginAvgDataLayerAccessPerTimeIntervalNb*: The average number of data layer accesses from a given plugin for a given workflow in a given time interval.
 - Required Input: Workflow name (W), Plugin name (P), Time Interval (T)



- Type: NbEventsPerTimeUnit (float)
 - Tags: Workflow, Plugin
 - Measurement Unit: –
 - Formula: $\frac{Count_{W,P,T}(PluginDataLayerAccess)}{WorkflowsPerTimeIntervalNb * WorkflowPluginTypeNb}$
4. *PlatformPluginAvgDataLayerAccessPerTimeIntervalNb*: The average number of data layer accesses from a given plugin in a given time interval, in all the workflows.
- Required Input: Plugin name (P), Time Interval (T)
 - Type: NbEventsPerTimeUnit (float)
 - Tags: Plugin
 - Measurement Unit: –
 - Formula: $\frac{Count_{P,T}(PluginDataLayerAccess)}{PluginInstanceCreationPerTimeIntervalNb}$
5. *WorkflowPluginTotalDataLayerAccessPerTimeIntervalDim*: The total size of the returned data from the data layer accesses, from a given plugin, for a given workflow, in a given time interval.
- Required Input: Workflow name (W), Plugin name (P), Time Interval (T)
 - Type: TotalDimension (integer)
 - Tags: Workflow, Plugin
 - Measurement Unit: bytes
 - Formula: $\sum_{W,P,T}(PluginDataLayerAccess)$
6. *PlatformPluginTotalDataLayerAccessPerTimeIntervalDim*: The total size of the returned data from the data layer accesses, from a given plugin, in a given time interval, in all the workflows.
- Required Input: Plugin name (P), Time Interval (T)
 - Type: TotalDimension (integer)
 - Tags: Plugin
 - Measurement Unit: bytes
 - Formula: $\sum_{P,T}(PluginDataLayerAccess)$
7. *WorkflowPluginAvgDataLayerAccessPerTimeIntervalDim*: The average size of the returned data from the data layer accesses, from a given plugin, for a given workflow, in a given time interval.
- Required Input: Workflow name (W), Plugin name (P), Time Interval (T)
 - Type: AvgDimension (float)
 - Tags: Workflow, Plugin
 - Measurement Unit: bytes
 - Formula: $\frac{WorkflowPluginTotalDataLayerAccessPerTimeIntervalDim}{WorkflowPluginTotalDataLayerAccessPerTimeIntervalNb}$



8. *PlatformPluginAvgDataLayerAccessPerTimeIntervalDim*: The average size of the returned data from the data layer accesses, from a given plugin in a given time interval, in all the workflows.

- Required Input: Plugin name (P), Time Interval (T)
- Type: AvgDimension (float)
- Tags: Plugin
- Measurement Unit: bytes
- Formula: $\frac{PlatformPluginTotalDataLayerAccessPerTimeIntervalDim}{PlatformPluginTotalDataLayerAccessPerTimeIntervalNb}$

4.2.10 Plugin related metrics - Cache Hit and Miss

1. *WorkflowPluginTotalCacheHitPerTimeIntervalNb*: The total number of cache hits that occurred inside a given plugin, in a given workflow, in a given time interval.

- Required Input: Workflow name (W), Plugin name (P), Time Interval (T)
- Type: TotalDimension (integer)
- Tags: Workflow, Plugin
- Measurement Unit: –
- Formula: $Count_{W,P,T}(PluginCacheHit)$

2. *PlatformPluginTotalCacheHitPerTimeIntervalNb*: The total number of cache hits that occurred inside a given plugin, in a given time interval.

- Required Input: Plugin name (P), Time Interval (T)
- Type: TotalDimension (integer)
- Tags: Plugin
- Measurement Unit: –
- Formula: $Count_{P,T}(PluginCacheHit)$

3. *WorkflowPluginTotalCacheMissPerTimeIntervalNb*: The total number of cache misses that occurred inside a given plugin, in a given workflow, in a given time interval.

- Required Input: Workflow name (W), Plugin name (P), Time Interval (T)
- Type: TotalDimension (integer)
- Tags: Workflow, Plugin
- Measurement Unit: –
- Formula: $Count_{W,P,T}(PluginCacheMiss)$

4. *PlatformPluginTotalCacheMissPerTimeIntervalNb*: The total number of cache misses that occurred inside a given plugin, in a given time interval.

- Required Input: Plugin name (P), Time Interval (T)
- Type: TotalDimension (integer)



- Tags: Plugin
 - Measurement Unit: –
 - Formula: $Count_{P,T}(PluginCacheMiss)$
5. *WorkflowPluginCacheHitRatePerTimeInterval*: The cache hit rate of a given plugin, in a given workflow, in a given time interval.
- Required Input: Workflow (W), Plugin name (P), Time Interval (T)
 - Type: TotalDimension (float)
 - Tags: Workflow, Plugin
 - Measurement Unit: –
 - Formula: $\frac{WorkflowPluginTotalCacheHitPerTimeIntervalNb}{WorkflowPluginTotalDataLayerAccessPerTimeIntervalNb} * 100$
6. *PlatformPluginCacheHitRatePerTimeInterval*: The cache hit rate of a given plugin, in a given time interval.
- Required Input: Plugin name (P), Time Interval (T)
 - Type: TotalDimension (float)
 - Tags: Plugin
 - Measurement Unit: –
 - Formula: $\frac{PlatformPluginTotalCacheHitPerTimeIntervalNb}{PlatformPluginTotalDataLayerAccessPerTimeIntervalNb} * 100$
7. *WorkflowPluginCacheMissRatePerTimeInterval*: The cache miss rate of a given plugin, in a given workflow, in a given time interval.
- Required Input: Workflow (W), Plugin name (P), Time Interval (T)
 - Type: TotalDimension (float)
 - Tags: Workflow, Plugin
 - Measurement Unit: –
 - Formula: $\frac{WorkflowPluginTotalCacheMissPerTimeIntervalNb}{WorkflowPluginTotalDataLayerAccessPerTimeIntervalNb} * 100$
8. *PlatformPluginCacheMissRatePerTimeInterval*: The cache miss rate of a given plugin, in a given time interval.
- Required Input: Plugin name (P), Time Interval (T)
 - Type: TotalDimension (float)
 - Tags: Plugin
 - Measurement Unit: –
 - Formula: $\frac{PlatformPluginTotalCacheMissPerTimeIntervalNb}{PlatformPluginTotalDataLayerAccessPerTimeIntervalNb} * 100$



4.2.11 Platform compound metrics

1. *PlatformAvgCPULoadPerTimeInterval*: The average CPU load of the platform in a given time interval.
 - Required Input: Time interval (T)
 - Type: AvgDimension (float)
 - Tags: No tags
 - Measurement Unit: percent
 - Formula: $\frac{\sum_T(PlatformCPULoad)}{Count_T(PlatformCPULoad)}$
2. *PlatformAvgMemoryUsedPerTimeInterval*: The average memory percent used by the platform in a given time interval.
 - Required Input: Time interval (T)
 - Type: AvgDimension (float)
 - Tags: No tags
 - Measurement Unit: percent
 - Formula: $\frac{\sum_T(PlatformMemoryUsage)}{Count_T(PlatformMemoryUsage)}$
3. *PlatformAvgMemoryDimPerTimeInterval*: The average memory dimension in bytes used by the platform in a given time interval.
 - Required Input: Time interval (T)
 - Type: AvgDimension (float)
 - Tags: No tags
 - Measurement Unit: bytes
 - Formula: $\frac{\sum_T(PlatformMemoryDim)}{Count_T(PlatformMemoryDim)}$
4. *WorkflowAvgNumberOfResultsPerTimeInterval*: The average number of results, produced by a given workflow in a given time interval
 - Required Input: Workflow name (W), Time interval (T)
 - Type: AvgDimension (float)
 - Tags: Workflow
 - Measurement Unit: –
 - Formula: $\frac{\sum_{W,T}(QueryResultInTriples)}{WorkflowsPerTimeIntervalNb}$
5. *PlatformAvgNumberOfResultsPerTimeInterval*: The average number of results, produced by the platform in a given time interval
 - Required Input: Time interval (T)
 - Type: AvgDimension (float)
 - Tags: No tags
 - Measurement Unit: –
 - Formula: $\frac{\sum_T(QueryResultInTriples)}{Count_TWorkflowInvocation}$



4.3 Generic metrics

4.3.1 Method related metrics

The generic metrics in this category capture the information about a method invocation, information that is obtained in a direct mode.

We will consider the following:

1. *WallClockTime*: elapsed time between method entry and method exit.
 - Type: DURATION (integer)
 - Tags: –
 - Measurement unit: microseconds
2. *ThreadUserCPUTime*: user CPU time spent by current thread executing this method.
 - Type: DURATION (integer)
 - Tags: –
 - Measurement unit: microseconds
3. *ThreadSystemCPUTime*: system CPU time spent by current thread executing this method.
 - Type: DURATION (integer)
 - Tags: –
 - Measurement unit: microseconds
4. *ThreadTotalCPUTime*: total CPU time spent by current thread executing this method.
 - Required Input: ThreadUserCPUTime, ThreadSystemCPUTime
 - Type: DURATION (integer)
 - Tags: –
 - Measurement unit: microseconds
 - Formula: $ThreadUserCPUTime + ThreadSystemCPUTime$
5. *ProcessTotalCPUTime*: total CPU time spent by current process (all threads from the application) executing this method.
 - Type: DURATION (integer)
 - Tags: –
 - Measurement unit: microseconds
6. *ThreadCount*: how many threads did this method invocation create.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: –
 - Measurement unit: –



7. *ThreadBlockCount*: the total number of times that the current thread executing this method entered the BLOCKED state.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: –
 - Measurement unit: –
8. *ThreadBlockTime*: the total accumulated time the current thread executing this method has been in the BLOCKED.
 - Type: DURATION (integer)
 - Tags: –
 - Measurement unit: microseconds
9. *ThreadWaitCount*: the total number of times that the current thread executing this method entered the WAITING or TIMED_WAITING state.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: –
 - Measurement unit: –
10. *ThreadWaitTime*: the total accumulated time the current thread executing this method has been in the WAITING or TIMED_WAITING state.
 - Type: DURATION (integer)
 - Tags: –
 - Measurement unit: microseconds
11. *GCCCount*: total number of collections that have occurred while executing this method.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: –
 - Measurement unit: –
12. *GCCTime*: approximate accumulated collection elapsed time in milliseconds while executing this method.
 - Type: DURATION (integer)
 - Tags: –
 - Measurement unit: microseconds

4.3.2 JVM related metrics

The generic metrics in this category capture the information that can be measured by instrumentation in direct mode per JVM by sampling, every X seconds for example.

We will consider the following:

1. *ClassLoadCount*: number of classes that are currently loaded in the Java virtual machine.



- Type: NUMBER_OF_INSTANCES (integer)
 - Tags: –
 - Measurement unit: –
2. *AllClassLoadCount*: total number of classes that have been loaded since the Java virtual machine has started execution.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: –
 - Measurement unit: –
 3. *AllClassUnloadCount*: total number of classes unloaded since the Java virtual machine has started execution.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: –
 - Measurement unit: –
 4. *CompilationTime*: approximate cumulated elapsed time (in milliseconds) spent in compilation.
 - Type: DURATION (integer)
 - Tags: –
 - Measurement unit: microseconds
 5. *GCCCount*: total number of collections that have occurred.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: –
 - Measurement unit: –
 6. *GCCTime*: approximate accumulated collection elapsed time in milliseconds.
 - Type: DURATION (integer)
 - Tags: –
 - Measurement unit: microseconds

4.3.3 System related metrics

The generic metrics in this category are those metrics that can be measured by the agent in direct mode by sampling, every number of seconds, the system or node information.

We will consider the following:

1. *ClassLoadCount*: number of classes that are currently loaded in the Java virtual machine.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: –



- Measurement unit: –
2. *AverageSystemLoad*: system load average for the a given period of time.
 - Required Input: System Load (S) at a given moment in time, Time Interval (T)
 - Type: LOAD (float)
 - Tags: –
 - Measurement unit: percent
 - Formula: $\frac{\sum_{S,T}(SystemLoad)}{SystemLoadMeasurementNb}$
 3. *TotalSystemFreeMemory*: the amount of total memory free in the system.
 - Type: DATA_SIZE (integer)
 - Tags: no
 - Measurement unit: bytes
 4. *TotalSystemUsedMemory*: the amount of total memory used in the system.
 - Type: DATA_SIZE (integer)
 - Tags: no
 - Measurement unit: bytes
 5. *SystemUsedSwapSpace*: the amount of swap space used by the system.
 - Type: DATA_SIZE (integer)
 - Tags: no
 - Measurement unit: bytes
 6. *SystemOpenFileDescriptor*: total number of system open file descriptors.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: –
 - Measurement unit: –
 7. *SwapIn*: Amount of memory swapped in from the disk.
 - Type: DATA_SIZE (integer)
 - Tags: –
 - Measurement unit: bytes
 8. *SwapOut*: Amount of memory swapped to the disk.
 - Type: DATA_SIZE (integer)
 - Tags: –
 - Measurement unit: bytes
 9. *IOIn*: Blocks sent to a block device.



- Type: DATA_SIZE (integer)
 - Tags: –
 - Measurement unit: –
10. *IOOut*: Blocks received from a block device.
- Type: DATA_SIZE (integer)
 - Tags: –
 - Measurement unit: –
11. *SystemInterrupts*: Total number of system interrupts.
- Type: NUMBER_OF_INSTANCES (integer)
 - Tags: –
 - Measurement unit: –
12. *SystemContextSwitchesCount*: Total number of system context switches.
- Type: NUMBER_OF_INSTANCES (integer)
 - Tags: –
 - Measurement unit: –
13. *UserCPULoad*: The User CPU LOAD of the node/system where the agent is running at a given time-stamp.
- Type: CPU_LOAD (float)
 - Tags: –
 - Measurement unit: percent
14. *SystemCPULoad*: The System CPU LOAD of the node/system where the agent is running at a given time-stamp.
- Type: CPU_LOAD (float)
 - Tags: –
 - Measurement unit: percent
15. *IdleLoad*: Time spent idle on the node/system where the agent is running at a given time-stamp.
- Type: CPU_LOAD (float)
 - Tags: –
 - Measurement unit: percent
16. *WaitLoad*: Time spent waiting for IO on the node/system where the agent is running at a given time-stamp.
- Type: CPU_LOAD (float)
 - Tags: –
 - Measurement unit: percent



4.3.4 Instrumented application related metrics

The generic metrics in this category are those metrics that are measured when the application or the agents are starting.

We will consider the following:

1. *OSName*: the name of the Operating System.
 - Type: VALUE (string)
 - Tags: –
 - Measurement unit: –
2. *OSVersion*: the version of the Operating System.
 - Type: VALUE (string)
 - Tags: –
 - Measurement unit: –
3. *IP Address*: the IP address. For multiple IP address on the same machine we will record all of them.
 - Type: VALUE (string)
 - Tags: –
 - Measurement unit: –
4. *CPUCount*: number of available CPU units.
 - Type: NUMBER_OF_INSTANCES (integer)
 - Tags: –
 - Measurement unit: –
5. *TotalMemory*: the size of total memory.
 - Type: DATA_SIZE (integer)
 - Tags: –
 - Measurement unit: bytes
6. *TotalSwap*: the size of total swap.
 - Type: DATA_SIZE (integer)
 - Tags: –
 - Measurement unit: bytes
7. *ApplicationName*: the name of the application.
 - Type: VALUE (string)
 - Tags: –
 - Measurement unit: –
8. *VMName*: the name of the java virtual machine.



- Type: VALUE (string)
- Tags: –
- Measurement unit: –

9. *VMVersion*: the version of the java virtual machine.

- Type: VALUE (string)
- Tags: –
- Measurement unit: –



5. Ontology of metrics

This chapter contains the formalization of the instrumentation and monitoring metrics introduced in Chapter 4 as an OWL ontology. The purpose of this ontology is to formally define each individual metric, atomic or compound, and to specify relations between them. The metrics are formalized as OWL classes, object and data properties being used to specify additional information about them. During monitoring, instances of OWL classes corresponding to metrics, will be created to represent measurement data collected from the plugins, workflows and platform. The aggregation of the monitoring measurements will be performed on the server side and all instance data (simple and aggregated) will be published into the LarkC Data Layer thus enabling reasoning over the monitored data.

The following listing contains the ontology in OWL. The full version of the ontology is available online at <http://wiki.larkc.eu/LarkcProject/WP11/IMOntology.owl>

Listing 5.1: Instrumentation and Monitoring Ontology - fragment

```

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix : <http://www.larkc.eu/ontologies/IMOntology.rdf#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@base <http://www.larkc.eu/ontologies/IMOntology.rdf> .

<http://www.larkc.eu/ontologies/IMOntology.rdf> rdf:type owl:Ontology
.

#####
#
#   Annotation properties
#
#####

<http://purl.org/dc/elements/1.1/description> rdf:type owl:
    AnnotationProperty .

#####
#
#   Object Properties
#
#####

###   http://www.larkc.eu/ontologies/IMOntology.rdf#hasMeasurementUnit

:hasMeasurementUnit rdf:type owl:ObjectProperty ;
    <http://purl.org/dc/elements/1.1/description> "specifies the
        measurement unit of a metric" ;
    rdfs:range :MeasurementUnit ;
    rdfs:domain :Metric .

###   http://www.larkc.eu/ontologies/IMOntology.rdf#hasMetricType
    
```



```
:hasMetricType rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the type
    of a metric" ;
  rdfs:domain :Metric ;
  rdfs:range :MetricType .

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasNamespace

:hasNamespace rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the
    namespace of a query" ;
  rdfs:range :Namespace ;
  rdfs:domain :Query .

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasPlatformCPULoad

:hasPlatformCPULoad rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "the CPU LOAD of the
    LarKC platform at a given timestamp" ;
  rdfs:domain :Platform ;
  rdfs:range :PlatformCPULoad .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasPlatformExecutionTime

:hasPlatformExecutionTime rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "the time elapsed since
    the platform was started until its execution has ended" ;
  rdfs:domain :Platform ;
  rdfs:range :PlatformExecutionTime .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasPlatformMemoryDim

:hasPlatformMemoryDim rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the
    memory dimension used by a platform" ;
  rdfs:domain :Platform ;
  rdfs:range :PlatformMemoryDim .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasPlatformMemoryUsage

:hasPlatformMemoryUsage rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "the memory
    percent usage of the platform at a given timestamp" ;
  rdfs:domain :Platform ;
  rdfs:range :PlatformMemoryUsage .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasPluginDataLayerAccessDim

:hasPluginDataLayerAccessDim rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "the size in BYTES
    of the returned data after an access to the data layer, during
    the execution of the plugin" ;
  rdfs:domain :Plugin ;
  rdfs:range :PluginDataLayerAccessDim .

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasPluginInputSize
```



```
:hasPluginInputSize rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the size of
    a plugin input" ;
  rdfs:domain :Plugin ;
  rdfs:range :PluginInputSize .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasPluginInstanceCreation

:hasPluginInstanceCreation rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the creation
    of a new plugin" ;
  rdfs:domain :Plugin ;
  rdfs:range :PluginInstanceCreation .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasPluginOutputSize

:hasPluginOutputSize rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the size
    of a plugin output" ;
  rdfs:domain :Plugin ;
  rdfs:range :PluginOutputSize .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasPluginTotalExecutionTime

:hasPluginTotalExecutionTime rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the total
    execution time for a plugin" ;
  rdfs:domain :Plugin ;
  rdfs:range :PluginTotalExecutionTime .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasQueryCompletionStatus

:hasQueryCompletionStatus rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the
    completion status of a query" ;
  rdfs:domain :Query ;
  rdfs:range :QueryCompletionStatus .

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasQuerySize

:hasQuerySize rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the size of
    a query" ;
  rdfs:domain :Query ;
  rdfs:range :QuerySize .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasQueryTotalResponseTime

:hasQueryTotalResponseTime rdf:type owl:ObjectProperty ;
  rdfs:domain :Query ;
  rdfs:range :QueryTotalResponseTime .

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasResult
```



```
:hasResult rdf:type owl:ObjectProperty ;
  rdfs:domain :Query ;
  rdfs:range :Result .

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasTag

:hasTag rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies one or
  more tags for a metric - used to specify the metric context" ;
  rdfs:range :Tag ;
  rdfs:domain owl:Thing .

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasValue

:hasValue rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the
  value of a metric, i.e. value of a measurement" ;
  rdfs:domain :Metric .

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasVariable

:hasVariable rdf:type owl:ObjectProperty ;
<http://purl.org/dc/elements/1.1/description> "specifies a variable
  of a query" ;
rdfs:domain :Query ;
rdfs:range :Variable .

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasWorkflowCPULoad

:hasWorkflowCPULoad rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "specified the load
  of the CPU during the execution of the workflow" ;
  rdfs:domain :Workflow ;
  rdfs:range :WorkflowCPULoad .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasWorkflowDuration

:hasWorkflowDuration rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the
  duration of a workflow" ;
  rdfs:domain :Workflow ;
  rdfs:range :WorkflowDuration .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasWorkflowMemoryDim

:hasWorkflowMemoryDim rdf:type owl:ObjectProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the
  memory dimension used by a workflow" ;
  rdfs:domain :Workflow ;
  rdfs:range :WorkflowMemoryDim .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasWorkflowMemoryUsage

:hasWorkflowMemoryUsage rdf:type owl:ObjectProperty ;
  rdfs:domain :Workflow ;
  rdfs:range :WorkflowMemoryUsage .
```



```
### http://www.w3.org/2002/07/owl#topObjectProperty
owl:topObjectProperty rdf:type owl:ObjectProperty .

#####
#
# Data properties
#
#####

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasContent
:hasContent rdf:type owl:DatatypeProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the
  string representation of the query" ;
  rdfs:domain :Query ;
  rdfs:range xsd:string .

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasDataValue
:hasDataValue rdf:type owl:DatatypeProperty .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasNumberOfDataSetSources

:hasNumberOfDataSetSources rdf:type owl:DatatypeProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the number
  of dataset source of FROM clauses in a query" ;
  rdfs:domain :Query ;
  rdfs:range xsd:integer .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasNumberOfNamespaces

:hasNumberOfNamespaces rdf:type owl:DatatypeProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the
  number of namespaces that are used in a query" ;
  rdfs:domain :Query ;
  rdfs:range xsd:integer .

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasNumberOfNodes

:hasNumberOfNodes rdf:type owl:DatatypeProperty ;
  <http://purl.org/dc/elements/1.1/description> "the number of
  nodes that the plugin is executed on" ;
  rdfs:domain :Plugin ;
  rdfs:range xsd:integer .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasNumberOfOperators

:hasNumberOfOperators rdf:type owl:DatatypeProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the
  number of logical operator in a query" ;
  rdfs:domain :Query ;
```



```
    rdfs:range xsd:integer .

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasNumberOfPlugins

:hasNumberOfPlugins rdf:type owl:DatatypeProperty ;
  <http://purl.org/dc/elements/1.1/description> "the total number of
  plugins that were involved in the execution of the workflow" ;
  rdfs:domain :Workflow ;
  rdfs:range xsd:integer .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasNumberOfResultLimit

:hasNumberOfResultLimit rdf:type owl:DatatypeProperty ;
  <http://purl.org/dc/elements/1.1/description> "gives the value
  of n from the SPARQL statement LIMIT n" ;
  rdfs:domain :Query ;
  rdfs:range xsd:integer .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasNumberOfResultOrdering

:hasNumberOfResultOrdering rdf:type owl:DatatypeProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the number
  of fields in the ORDER BY statement" ;
  rdfs:domain :Query ;
  rdfs:range xsd:integer .

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasNumberOfThreads

:hasNumberOfThreads rdf:type owl:DatatypeProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the total
  number of threads started during the execution of the workflow"
  ;
  rdfs:domain :Workflow ;
  rdfs:range xsd:integer .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasNumberOfThreadsInPlugin

:hasNumberOfThreadsInPlugin rdf:type owl:DatatypeProperty ;
  <http://purl.org/dc/elements/1.1/description> "the number of
  threads started by the plugin" ;
  rdfs:domain :Plugin ;
  rdfs:range xsd:integer .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasNumberOfVariables

:hasNumberOfVariables rdf:type owl:DatatypeProperty ;
  <http://purl.org/dc/elements/1.1/description> "specifies the
  number of variables that are used in a query" ;
  rdfs:domain :Query ;
  rdfs:range xsd:integer .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  hasNumberResultOffset

:hasNumberResultOffset rdf:type owl:DatatypeProperty ;
```



```
<http://purl.org/dc/elements/1.1/description> "gives the value
  of m from the SPARQL statement OFFSET m" ;
rdfs:domain :Query ;
rdfs:range xsd:integer .

### http://www.larkc.eu/ontologies/IMOntology.rdf#hasTimeStamp

:hasTimeStamp rdf:type owl:DatatypeProperty ;
<http://purl.org/dc/elements/1.1/description> "specifies the time
  stamp when the measurement was performed" ;
rdfs:domain :Metric ;
rdfs:range xsd:dateTime .

### http://www.w3.org/2002/07/owl#topDataProperty

owl:topDataProperty rdf:type owl:DatatypeProperty .

#####
#
# Classes
#
#####

### http://www.larkc.eu/ontologies/IMOntology.rdf#CPULoad

:CPULoad rdf:type owl:Class ;
rdfs:subClassOf :MetricType ;
<http://purl.org/dc/elements/1.1/description> "The load of the
  CPU at the time of measurement (per computation node)" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#Duration

:Duration rdf:type owl:Class ;
rdfs:subClassOf :MetricType ;
<http://purl.org/dc/elements/1.1/description> "The time between
  two events (e.g. the time it took for some code to be
  executed)" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#Event

:Event rdf:type owl:Class ;
rdfs:subClassOf :MetricType ;
<http://purl.org/dc/elements/1.1/description> "Some code point has
  been reached and the time stamp at which this event occurred is
  transmitted" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#Exception

:Exception rdf:type owl:Class ;
rdfs:subClassOf :MetricType ;
<http://purl.org/dc/elements/1.1/description> "An exceptional
  condition occurred while processing the request" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#MeasurementUnit

:MeasurementUnit rdf:type owl:Class ;
```



```
<http://purl.org/dc/elements/1.1/description> "concept that
  represents an abstract measurement unit" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#MemoryPercent

:MemoryPercent rdf:type owl:Class ;
  rdfs:subClassOf :MetricType ;
  <http://purl.org/dc/elements/1.1/description> "The memory load at
    the time of the measurement (per computation node)" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#Metric

:Metric rdf:type owl:Class ;
  <http://purl.org/dc/elements/1.1/description> "concept that
    represents an abstract metric" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#MetricType

:MetricType rdf:type owl:Class .

### http://www.larkc.eu/ontologies/IMOntology.rdf#Namespace

:Namespace rdf:type owl:Class ;
  rdfs:subClassOf [ rdf:type owl:Restriction ;
    owl:onProperty :hasDataValue ;
    owl:someValuesFrom xsd:anyURI
  ] ;
  <http://purl.org/dc/elements/1.1/description> "concept that
    represents a namespace from a SPARQL query" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#Platform

:Platform rdf:type owl:Class ;
  <http://purl.org/dc/elements/1.1/description> "concept that
    represents a LarKC platform" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#PlatformCPULoad

:PlatformCPULoad rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMetricType ;
    owl:allValuesFrom :CPULoad
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMeasurementUnit ;
    owl:hasValue :Percent
  ] ;
  <http://purl.org/dc/elements/1.1/description> "the CPU LOAD of
    the LarKC platform at a given timestamp" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
PlatformExecutionTime

:PlatformExecutionTime rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMeasurementUnit ;
    owl:hasValue :Microsecond
```



```

] ,
[ rdf:type owl:Restriction ;
  owl:onProperty :hasDataValue ;
  owl:allValuesFrom xsd:long
] ,
[ rdf:type owl:Restriction ;
  owl:onProperty :hasMetricType ;
  owl:allValuesFrom :Duration
] ;
  <http://purl.org/dc/elements/1.1/description> "the time elapsed
    since the platform was started until its execution has ended"
  .

### http://www.larkc.eu/ontologies/IMOntology.rdf#PlatformMemoryDim

:PlatformMemoryDim rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMetricType ;
    owl:allValuesFrom :Size
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMeasurementUnit ;
    owl:hasValue :Byte
  ] ;
  <http://purl.org/dc/elements/1.1/description> "the amount of
    memory used by the platform at a given time-stamp" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
PlatformMemoryUsage

:PlatformMemoryUsage rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMetricType ;
    owl:allValuesFrom :MemoryPercent
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMeasurementUnit ;
    owl:hasValue :Percent
  ] ;
  <http://purl.org/dc/elements/1.1/description> "the memory percent
    usage of the platform at a given time-stamp" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#Plugin

:Plugin rdf:type owl:Class ;
  <http://purl.org/dc/elements/1.1/description> "concept that
    represents a LarKC plugin" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
PluginDataLayerAccessDim

:PluginDataLayerAccessDim rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Class ;
    owl:intersectionOf ( [ rdf:type owl:Restriction ;
    owl:onProperty :hasTag ;

```



```

    owl:hasValue :PluginTag
  ]
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasTag ;
    owl:hasValue :QueryTag
  ]
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasTag ;
    owl:hasValue :WorkflowTag
  ]
)
] ,
[ rdf:type owl:Restriction ;
  owl:onProperty :hasMetricType ;
  owl:allValuesFrom :Size
] ;

<http://purl.org/dc/elements/1.1/description> "the size in BYTES of
the returned data after an access to the data layer, during the
execution of the plugin" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#PluginException

:PluginException rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Class ;
    owl:intersectionOf ( [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :PluginTag
    ]
    [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :QueryTag
    ]
    [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :WorkflowTag
    ]
  )
] ,
[ rdf:type owl:Restriction ;
  owl:onProperty :hasMetricType ;
  owl:allValuesFrom :Exception
] ;

<http://purl.org/dc/elements/1.1/description> "marks an uncaught
exception in a plugin" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#PluginInputSize

:PluginInputSize rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Class ;
    owl:intersectionOf ( [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :PluginTag
    ]
    [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;

```



```

        owl:hasValue :QueryTag
      ]
      [ rdf:type owl:Restriction ;
        owl:onProperty :hasTag ;
        owl:hasValue :WorkflowTag
      ]
    )
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasDataValue ;
    owl:allValuesFrom xsd:integer
  ] ;
  <http://purl.org/dc/elements/1.1/description> "concept that
    represents the size of the input of a plugin" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  PluginInputSizeInBytes

:PluginInputSizeInBytes rdf:type owl:Class ;
  rdfs:subClassOf :PluginInputSize ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMeasurementUnit ;
    owl:hasValue :Byte
  ] ;
  <http://purl.org/dc/elements/1.1/description> "the size of the
    input of a plugin represented in bytes" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  PluginInputSizeInTriples

:PluginInputSizeInTriples rdf:type owl:Class ;
  rdfs:subClassOf :PluginInputSize ;
  <http://purl.org/dc/elements/1.1/description> "the size of the input
    of a plugin represented in number of RDF triples" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  PluginInstanceCreation

:PluginInstanceCreation rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Class ;
    owl:intersectionOf ( [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :PluginTag
    ]
      [ rdf:type owl:Restriction ;
        owl:onProperty :hasTag ;
        owl:hasValue :QueryTag
      ]
      [ rdf:type owl:Restriction ;
        owl:onProperty :hasTag ;
        owl:hasValue :WorkflowTag
      ]
    )
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMetricType ;
    owl:allValuesFrom :Event
  ] ;

```



```

    <http://purl.org/dc/elements/1.1/description> "the
      instantiation of a new plugin" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  PluginOutputSizeInBytes

:PluginOutputSizeInBytes rdf:type owl:Class ;
  rdfs:subClassOf :PluginOutputSize ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMeasurementUnit ;
    owl:hasValue :Byte
  ] ;
  <http://purl.org/dc/elements/1.1/description> "the size of the
    output of a plugin represented in bytes" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#PluginOutputSize

:PluginOutputSize rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasDataValue ;
    owl:allValuesFrom xsd:integer
  ] ,
  [ rdf:type owl:Class ;
    owl:intersectionOf ( [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :PluginTag
    ]
    [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :QueryTag
    ]
    [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :WorkflowTag
    ]
  )
  ] ;
  <http://purl.org/dc/elements/1.1/description> "concept that
    represents the size of the output of a plugin" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  PluginOutputSizeInTriples

:PluginOutputSizeInTriples rdf:type owl:Class ;
  rdfs:subClassOf :PluginOutputSize ;
  <http://purl.org/dc/elements/1.1/description> "the size of the
    output of a plugin represented in number of RDF triples" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  PluginTotalExecutionTime

:PluginTotalExecutionTime rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMeasurementUnit ;
    owl:hasValue :Microsecond
  ] ,

```



```

    [ rdf:type owl:Restriction ;
      owl:onProperty :hasDataValue ;
      owl:allValuesFrom xsd:integer
    ] ,
    [ rdf:type owl:Class ;
      owl:intersectionOf ( [ rdf:type owl:Restriction ;
        owl:onProperty :hasTag ;
        owl:hasValue :PluginTag
      ]
    [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :QueryTag
    ]
    [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :WorkflowTag
    ]
      )
    ] ;
<http://purl.org/dc/elements/1.1/description> "denotes the total
  execution time of a plugin inside the workflow" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#Query

:Query rdf:type owl:Class ;
  <http://purl.org/dc/elements/1.1/description> "concept that
    represents a SPARQL query" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  QueryCompletionStatus

:QueryCompletionStatus rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
[ rdf:type owl:Restriction ;
  owl:onProperty :hasTag ;
  owl:hasValue :QueryTag
] ,
[ rdf:type owl:Restriction ;
  owl:onProperty :hasDataValue ;
  owl:someValuesFrom xsd:boolean
] .

### http://www.larkc.eu/ontologies/IMOntology.rdf#QueryInvocation

:QueryInvocation rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMetricType ;
    owl:allValuesFrom :Event
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasTag ;
    owl:hasValue :QueryTag
  ] ;
  <http://purl.org/dc/elements/1.1/description> "records the fact
    that a new query has been sent to the platform" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#QuerySize

```



```

:QuerySize rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasDataValue ;
    owl:someValuesFrom xsd:integer
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasTag ;
    owl:hasValue :QueryTag
  ] ;
  <http://purl.org/dc/elements/1.1/description> "concept that
  represents the size of a query" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#QuerySizeInBytes

:QuerySizeInBytes rdf:type owl:Class ;
  rdfs:subClassOf :QuerySize ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMeasurementUnit ;
    owl:hasValue :Byte
  ] ;
  <http://purl.org/dc/elements/1.1/description> "the size of a
  query represented in bytes" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#QuerySizeInTriples

:QuerySizeInTriples rdf:type owl:Class ;
  rdfs:subClassOf :QuerySize ;
  <http://purl.org/dc/elements/1.1/description> "the size of a query
  represented in terms of number of RDF triples" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
  QueryTotalResponseTime

:QueryTotalResponseTime rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMeasurementUnit ;
    owl:hasValue :Microsecond
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasTag ;
    owl:hasValue :QueryTag
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasDataValue ;
    owl:someValuesFrom xsd:integer
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMetricType ;
    owl:allValuesFrom :Duration
  ] ;
  <http://purl.org/dc/elements/1.1/description> "denotes the time
  elapsed from when the query was sent to the platform until
  the final result is provided to the user" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#Result

:Result rdf:type owl:Class ;

```



```
    rdfs:subClassOf owl:Thing ;
    <http://purl.org/dc/elements/1.1/description> "concept that
        represents the result of a SPARQL query" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#ResultSize

:ResultSize rdf:type owl:Class ;
    rdfs:subClassOf :Metric ,
    [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :QueryTag
    ] ,
    [ rdf:type owl:Restriction ;
      owl:onProperty :hasDataValue ;
      owl:someValuesFrom xsd:integer
    ] ;
    <http://purl.org/dc/elements/1.1/description> "concept that
        represents the size of a result" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#ResultSizeInBytes

:ResultSizeInBytes rdf:type owl:Class ;
    rdfs:subClassOf :ResultSize ;
    <http://purl.org/dc/elements/1.1/description> "the size of a
        result represented in bytes" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#
    ResultSizeInTriples

:ResultSizeInTriples rdf:type owl:Class ;
    rdfs:subClassOf :ResultSize ;
    <http://purl.org/dc/elements/1.1/description> "the size of a
        result represented in terms of number of RDF triples" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#Size

:Size rdf:type owl:Class ;
    rdfs:subClassOf :MetricType ;
    <http://purl.org/dc/elements/1.1/description> "The size of the
        data structure" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#Tag

:Tag rdf:type owl:Class ;
    owl:equivalentClass [ rdf:type owl:Class ;
      owl:oneOf ( :WorkflowTag
        :QueryTag
        :ThreadTag
        :NodeTag
        :MethodTag
        :PluginTag
      )
    ] .

### http://www.larkc.eu/ontologies/IMOntology.rdf#Variable

:Variable rdf:type owl:Class ;
    <http://purl.org/dc/elements/1.1/description> "concept that
        represents a variable from a SPARQL query" .
```



```
### http://www.larkc.eu/ontologies/IMOntology.rdf#Workflow

:Workflow rdf:type owl:Class ;
  <http://purl.org/dc/elements/1.1/description> "concept that
    represents a LarKC workflow" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#WorkflowCPULoad

:WorkflowCPULoad rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Class ;
    owl:intersectionOf ( [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :QueryTag
    ]
    [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :WorkflowTag
    ]
  )
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMetricType ;
    owl:allValuesFrom :CPULoad
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMeasurementUnit ;
    owl:hasValue :Percent
  ] ;
  <http://purl.org/dc/elements/1.1/description> "represents the
    load of the CPU during the execution of the workflow" .

### http://www.larkc.eu/ontologies/IMOntology.rdf#WorkflowDuration

:WorkflowDuration rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Class ;
    owl:unionOf ( [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :QueryTag
    ]
    [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :WorkflowTag
    ]
  )
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMetricType ;
    owl:allValuesFrom :Duration
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasDataValue ;
    owl:someValuesFrom xsd:integer
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMeasurementUnit ;
    owl:hasValue :Microsecond
  ] .
```



```
### http://www.larkc.eu/ontologies/IMOntology.rdf#WorkflowMemoryDim
```

```
:WorkflowMemoryDim rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Class ;
    owl:intersectionOf ( [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :QueryTag
    ]
    [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :WorkflowTag
    ]
  )
] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMeasurementUnit ;
    owl:hasValue :Byte
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMetricType ;
    owl:allValuesFrom :Size
  ] .
```

```
### http://www.larkc.eu/ontologies/IMOntology.rdf#
WorkflowMemoryUsage
```

```
:WorkflowMemoryUsage rdf:type owl:Class ;
  rdfs:subClassOf :Metric ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasMeasurementUnit ;
    owl:hasValue :Percent
  ] ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :hasDataValue ;
    owl:allValuesFrom xsd:integer
  ] ,
  [ rdf:type owl:Class ;
    owl:intersectionOf ( [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :QueryTag
    ]
    [ rdf:type owl:Restriction ;
      owl:onProperty :hasTag ;
      owl:hasValue :WorkflowTag
    ]
  )
] .
```

```
### http://www.w3.org/2002/07/owl#Thing
```

```
owl:Thing rdf:type owl:Class .
```

```
#####
#
#   Individuals
#
#####
```



```
### http://www.larkc.eu/ontologies/IMOntology.rdf#Byte
:Byte rdf:type :MeasurementUnit ,
      owl:NamedIndividual .

### http://www.larkc.eu/ontologies/IMOntology.rdf#MethodTag
:MethodTag rdf:type :Tag ,
           owl:NamedIndividual .

### http://www.larkc.eu/ontologies/IMOntology.rdf#Microsecond
:Microsecond rdf:type :MeasurementUnit ,
             owl:NamedIndividual .

### http://www.larkc.eu/ontologies/IMOntology.rdf#NodeTag
:NodeTag rdf:type :Tag ,
         owl:NamedIndividual .

### http://www.larkc.eu/ontologies/IMOntology.rdf#Percent
:Percent rdf:type :MeasurementUnit ,
         owl:NamedIndividual .

### http://www.larkc.eu/ontologies/IMOntology.rdf#PluginTag
:PluginTag rdf:type :Tag ,
          owl:NamedIndividual .

### http://www.larkc.eu/ontologies/IMOntology.rdf#QueryTag
:QueryTag rdf:type :Tag ,
         owl:NamedIndividual .

### http://www.larkc.eu/ontologies/IMOntology.rdf#ThreadTag
:ThreadTag rdf:type :Tag ,
         owl:NamedIndividual .

### http://www.larkc.eu/ontologies/IMOntology.rdf#WorkflowTag
:WorkflowTag rdf:type :Tag ,
            owl:NamedIndividual .
```



6. Conclusion

This deliverable reported on the work that has been done in LarKC towards identifying, modeling and formally representing, using semantic technologies, the metrics that will be considered for instrumentation and monitoring of LarKC plugins, workflows and platform. Instrumentation and monitoring are complex tasks that can be performed a different levels. We distinguished different levels of abstraction and we distributed each of the metrics to one or more of these levels. Furthermore the metrics were defined precisely in terms of what they represent, how they can be computed, what measurement unit they use, etc. We identified different dimensions that are relevant when classifying metrics. Metrics can be classified as either atomic, meaning that they can be measured directly or they can be compound and in this case the metrics are defined in relation to other metrics that they aggregate. Another way to classify the metrics, as described in the deliverable, is to focus on how specific or general a metric is. We distinguished between general metrics that apply to any information system and metrics that are specific to LarKC, its plugins, workflows and components.

Another core contribution of this deliverable is the ontology of metrics. The ontology provides a formal specification of the conceptualization of metrics domains. The idea of using an ontology to represent the metrics will allow reasoning of measurements and will enable more clever aggregation and visualization of the data. The purpose of this ontology is formally define each individual metric, atomic or compound, and to specify relations between them. The ontology of metrics is an OWL ontology that contains classes, object and data properties, and instances. During monitoring, instances of classes corresponding to metrics, will be created to represent measurement data collected from the plugins, workflows and platform. The definition and modeling of the metrics, and by consequence the ontology of metrics is a core part of the instrumentation and monitoring platform developed in WP11 and will be used directly in the first version of the instrumentation and monitoring platform, that will be described in deliverable D11.2, due M35.



REFERENCES

- [1] R. Brehar, I. Toma, S. Nedevschi, M. Negru, S. Bota, I. Giosan, A. Vatavu, C. Vicas, and M. Chezan, “State of the art and requirements analysis,” LarKC Project Deliverable, Tech. Rep. D11.1.1, 2010.