



## LarKC

*The Large Knowledge Collider*

*a platform for large scale integrated reasoning and Web-search*

FP7 – 215535

---

# D11.1.1 State of the art and requirements analysis

---

**Coordinator: Raluca Brehar (UTC)**

**With contributions from: Ioan Toma (Softgress), Sergiu Nedevschi(UTC), Mihai Negru (UTC), Silviu Bota (UTC), Ionel Giosan (UTC), Andrei Vatavu (UTC), Cristian Vicas (UTC), Mihai Chezan (Softgress)**

**Quality Assessor: Blaz Fortuna (CycCorp)**

**Quality Controller: Sergiu Nedevschi (UTC)**

Document Identifier:	LarKC/2008/D11.1.1/V1.0
Class Deliverable:	LarKC EU-IST-2008-215535
Version:	version 1.0.0
Date:	November 25, 2010
State:	final
Distribution:	public



## EXECUTIVE SUMMARY

In this document we identify the key requirements for instrumentation and monitoring of LarKC platform and plug-ins, and we investigate the state of the art in the field of instrumentation and monitoring. We introduce a set of typical scenarios for LarKC users (including plug-in and platform developers) who want to instrument and monitor the LarKC plug-ins and/or components. Based on these scenarios we identify a set of functional and non-functional requirements for each component of the instrumentation and monitoring solution. With these requirements in mind, we perform a survey and analysis of tools and technologies for instrumentation, monitoring, visualization, etc. The goal of the survey is to identify which of the existing tools and technologies can be used as part of the monitoring and instrumentation solution that we build in WP11.



## DOCUMENT INFORMATION

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

<b>Deliverable</b>	<b>Number</b>	11.1.1	<b>Title</b>	State of the art and requirements analysis
<b>Work Package</b>	<b>Number</b>	11	<b>Title</b>	Instrumentation and Monitoring







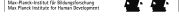





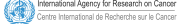

<b>Date of Delivery</b>	<b>Contractual</b>	M32	<b>Actual</b>	30-Nov-10
<b>Status</b>	version 1.0.0		final	<input checked="" type="checkbox"/>
<b>Nature</b>	prototype	<input type="checkbox"/>	report	<input checked="" type="checkbox"/>
<b>Dissemination Level</b>	public	<input checked="" type="checkbox"/>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

<b>Authors (Partner)</b>	UCT, Softgress			
<b>Resp. Author</b>	Raluca Brehar		<b>E-mail</b>	raluca.brehar@cs.utcluj.ro
	<b>Partner</b>	UTC	<b>Phone</b>	+40 264 401484



<b>Abstract (for dissemination)</b>	In this document we identify the key requirements for instrumentation and monitoring of LarKC platform and plug-ins, and we investigate the state of the art in the field of instrumentation and monitoring. We introduce a set of typical scenarios for LarKC users (including plug-in and platform developers) who want to instrument and monitor the LarKC plug-ins and/or components. Based on these scenarios we identify a set of functional and non-functional requirements for each component of the instrumentation and monitoring solution. With these requirements in mind, we perform a survey and analysis of tools and technologies for instrumentation, monitoring, visualization, etc. The goal of the survey is to identify which of the existing tools and technologies can be used as part of the monitoring and instrumentation solution that we build in WP11.
<b>Keywords</b>	State of the art, Instrumentation and Monitoring, Requirements, Tools and Technologies for Instrumentation and 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 <a href="http://www.utcluj.ro/">http://www.utcluj.ro/</a>	 The logo of the Technical University of Cluj-Napoca, featuring a stylized 'U' and 'T' in red and grey.	Prof. Dr. Eng. Sergiu Nedevschi TECHNICAL UNIVERSITY OF CLUJ-NAPOCA Cluj-Napoca, Romania E-mail: <a href="mailto:sergiu.nedevschi@cs.utcluj.ro">sergiu.nedevschi@cs.utcluj.ro</a>
SOFTGRESS S.R.L. <a href="http://www.softgress.com/">http://www.softgress.com/</a>	 The logo for Softgress, consisting of the word 'Softgress' in white text on a blue rectangular background.	Dr. Ioan Toma SOFTGRESS S.R.L. Cluj-Napoca, Romania E-mail: <a href="mailto:ioan.toma@softgress.com">ioan.toma@softgress.com</a>



# TABLE OF CONTENTS

LIST OF FIGURES	7
1 INTRODUCTION	8
2 SCENARIOS	9
2.1 Platform functionality	9
2.2 Resource utilization	10
2.3 Query history analysis	10
2.4 Work-flow statistics analysis	11
2.5 Scalability analysis	11
2.6 Bottleneck prediction	11
2.7 Work-flow prediction based on raw data	11
3 REQUIREMENTS	13
3.1 High level requirements and system components	13
3.2 System components requirements	14
3.2.1 Instrumentation requirements	14
3.2.2 Agent requirements	16
3.2.3 Server requirements	17
3.2.4 Visualization requirements	18
3.2.5 Relevance feedback requirements	19
4 STATE OF THE ART	21
4.1 Instrumentation	21
4.1.1 Java tools that can be used for code instrumentation	21
4.1.2 Java instrumentation and monitoring tools	23
4.2 Monitoring approaches - agents and server	25
4.3 Data visualization	29
4.3.1 Introduction	29
4.3.2 Visualization architecture and components	30
4.3.3 Web development frameworks	31
4.3.4 Data visualization tools	38
4.4 Relevance feedback	40
4.4.1 Data preprocessing	40
4.4.2 Model construction and deployment	43
4.4.3 Data mining for predicting system performance	45
4.4.4 Data mining tools	46
5 CONCLUSION	49



# LIST OF FIGURES

- 3.1 Instrumentation and monitoring architecture . . . . . 14
- 3.2 Relevance feedback component within the monitoring and instrumen-  
tation architecture . . . . . 20
  
- 4.1 Data visualization and reporting . . . . . 31
- 4.2 Steps in the data mining process . . . . . 41



## 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, and 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. Particularly, for LarKC, developers would like to know how performant their plugins and components 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 are know 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 measurement of various metrics that are of interest to the user. Monitoring, on the other hand, is the process of performing the measurements and making the system's user aware of the measurements.

The LarKC platform, as 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 platform's level, makes it 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 deliverable is to identify what are the key requirements for instrumentation and monitoring of the LarKC platform and plugins, and to investigate the current state of the art in the field of instrumentation and monitoring. We start by describing a set of typical scenarios for LarKC users (including plugin and platform developers) who want to instrument and monitor they LarKC plugin and / or components. Based on these scenarios we identify a set of functional and non-functional requirements for each component of the instrumentation and monitoring solution. With these requirements in mind, we then perform a systematic survey and analysis of tools and technologies for instrumentation, monitoring, visualization, etc. The goal of the survey is to identify which of the existing methods and tools can be used as part of the monitoring and instrumentation solution that we build in WP11. A default requirement for these tools is open source, license-compatiblity with existing LarKC platform components.

This deliverable is organized as follows. In Chapter 2, we describe a set of scenarios that capture how LarKC users, both plugin and platform/component developers, interact and use the platform and the plugins. In Chapter 3 we introduce first at a very generic level the high level requirements and components of the instrumentation and monitoring platform we build in WP11. These requirements are further detailed in the same chapter at the level of each component. Chapter 4 contains a detailed description of the state of the art of tools and technologies that can be potentially used to develop an instrumentation and monitoring solution for a distributed platform such as LarKC. Finally, Chapter 5 concludes the deliverable.



## 2. Scenarios

This chapter contains a set of typical scenarios for LarKC users (including plug-in and platform developers) who want to instrument and monitor the LarKC plug-in and/or components.

In the description of the initial operational framework for LarKC the following interaction patterns have been identified [1]:

- **Plug-in Construction:** One possible way to interact with the platform is as a plug-in designer/writer. During this phase, designers will write plug-ins and deploy them in the platform for future use.
- **Platform Configuration:** Another interaction scenario involves users (platform configuration designers) writing scripts to combine existing plug-ins in order to solve a given task. For example, users may write a Decider plug-in that combines Selection, Identify, Transform and Reasoning plug-ins to answer SPARQL queries.
- **Platform Invocation:** Finally, end users will be able to use the services provided by the platform, i.e. a particular plug-in configuration to execute SPARQL queries. This interaction pattern also includes application designers that implement applications that need to use the services provided by the platform.

Based on these patterns we have described some scenarios that demonstrate the role of the components designed for the instrumentation and monitoring of the LarKC platform.

### 2.1 Platform functionality

In this use case the users are provided statistics about:

- queries that have been run on the platform.  
Possible features surprised by these statistics may attain the following goals: amount of queries that have been asked in a time interval; percent of queries that finished with success (during a time interval); percent of queries that did not finish with success / failures during a time interval; number of queries processed in a time interval; amount of queries that have been asked simultaneously.
- data that is transmitted within the platform, between plug-ins.  
The aims related to data transmission could be: average data size transmitted to the platform in a time interval; average size of the input data for each type of plug-in identifier, transformer, selector, reasoner during a time interval; average dimension of the output data for each type of plug-in identifier, transformer, selector, reasoner during a time interval; average amount of the RDF triples transmitted to the data layer in a given time interval,
- plug-ins that have run within the platform.  
We may refer to the following: instances of a given plug-in are instantiated during a time interval; number of deciders / reasoners / selectors / identifiers / transformers that have been instantiated in a given time interval; number of deciders / reasoners / selectors / identifiers / transformers that have failed in a given time interval; amount of plug-ins of a given type (deciders / reasoners / selectors / identifiers / transformers) that are running in parallel .



- work-flows that have run within the platform.  
For this aspect of the use case we may show: how many different work-flows have been run during a time interval; what is the percentage of time spent in work-flow; how many work-flows are run in parallel.

## 2.2 Resource utilization

In this use case the users may analyze the system load, the resources used by the platform and the plug-ins. The targets of the proposed use case are:

- statistics about the JVM used by the platform and plug-ins. The acquired statistics may include: amount of memory consumed by the JVM, number of threads that are running in the JVM, percent of the available processing power that is being used by the JVM.
- plug-in behavior analysis: measures the performance of the plug-in at runtime on a given query. The performed analysis could include the time spend in the execution of a certain method in a given plug-in, the resources used by the execution of a certain method in the given plug-in, how many accesses to the data layer are made by a given plug-in, the size of the data that is received/sent by the given plug-in.
- plug-in in the work-flow analysis use case: measures the performance of a plug-in in the context of a given work-flow run on a given query. The investigations could compare the execution time of the work-flow, if it is run on a single machine or on a grid, show the percentage of time relative to the overall work-flow time spent in a given plug-in.
- plug-in in the platform analysis: offer statistics about the plug-in in the context of the platform and of thousands of end-users. The evaluation can comprise how many work-flows have used a given plug-in in the last days (in a given period of time), how many queries have been successfully answered by work-flows that contained a given plug-in, how many work-flows that used a given plug-in have failed, how much time took for the execution of a given plug-in in the last days on all the work-flows that have used it.

## 2.3 Query history analysis

In this scenario the user composes a query and wants to find all the historical information related to his/her query that was recorded in the database of the instrumentation tool. The targets for this use case could be:

- amount of queries that have used the same name-space as my query in the last days.
- number of queries that used similar name-spaces and similar work-flows (same plug-ins or most plug-ins are similar with the ones in my work-flow).
- average response time for the above queries.
- summary of the resources used by the above queries.



In our view, the classic interaction with the LarKC platform is a three step process that includes: (1) asking a query and specifying workflow parameters, (2) composing and running the workflow and (3) providing the results. This interaction pattern slightly changes when we consider streaming workflows. However, some of the atomic and generic metrics that we have previously identified in deliverable [2] apply for both types of interactions (streaming and non-streaming) and thus the instrumentation and monitoring solutions developed by WP11, including visualization, can be used in both cases.

## 2.4 Work-flow statistics analysis

In this use case the user composes a work-flow and wants to find all the historical information related to his/her work-flow that was recorded in the database of the instrumentation tool.

The statistical analysis could comprise:

- similar work-flows have been run in the last days (same plug-ins, or same decider etc).
- average response time for these work-flows.
- summary of the resources used by these work-flows.

## 2.5 Scalability analysis

Given a query and a work-flow predict the resources and execution time for a successful accomplishment of the query. We may include: the time needed until the platform will provide an answer to the given query, in the conditions of a work-flow; amount of memory that could be used for a successful execution of a given query ; number of interactions with the data layer; the average CPU load during the execution of the query/work-flow.

In this scenario we will do no inference or reasoning over the datasets contained in the query, we will not try to understand what important concepts are given in the query. We will just experiment data mining algorithms and try to predict the above mentioned parameters based only on the values of previous stored metrics. Hence, our predictions could be in some cases less accurate.

## 2.6 Bottleneck prediction

Predict if the platform will provide a result to the query and the work-flow will have a successful execution (success/failure analysis) or find the probability that work-flow chosen for my query to have a successful execution. For this scenario we plan to use the relevance feedback mechanism, that based on the success/error metrics will provide some suggestions about a possible failure in execution.

## 2.7 Work-flow prediction based on raw data

Given a query, predict possible work-flows to solve it. We could include all the work-flows that may solve a given query or all the work-flows that use a certain transformer,

selector, identifier or reasoner and may solve a specified query. This scenario will be solved via the relevance feedback mechanism. It is important to state here that the relevance feedback mechanism will not be a tool for the automatic composition of work-flows. It will experiment prediction algorithms and offer some suggestions for plug-ins, for a given query based on the data recorded in the query related metrics and considering the previous behavior of the platform, also recorded in atomic and compound metrics.



## 3. Requirements

### 3.1 High level requirements and system components

In this section we sketch the high level requirements and the components that will implement and fulfill these requirements as part of the instrumentation and monitoring solution. We distinguish between two types of requirements, namely functional and non-functional requirements. From the functional requirements we derive a set of components. Both are of similar importance and they have been extracted through a careful analysis of the scenarios described in the previous chapter. The functional requirements in particular are relevant for us, since we can group required functionalities and identify components of LarKC instrumentation and monitoring solution. The following functional requirements are relevant for the overall instrumentation and monitoring.

- the system should perform measurements of various metrics
- the system should collect measurements data and should aggregate it
- the system should allow querying of the monitored data
- the system should provide an intuitive visualization capabilities for real-time and historical data
- the system should be able to predict metrics values
- the system should propose and select the most appropriate configuration to run based on past learned experiences

The previous list is not intended to be an exhaustive list of requirements, but only to point out the most important high level functional requirements that will be further refined for each of instrumentation and monitoring components in Section 3.2. The following non-functional requirements are relevant for the overall instrumentation and monitoring.

- the system should be scalable, having a low response time
- the system should be customizable and easily configurable
- the system should be portable

As in the case of functional requirements, the previous list of non-functional requirements is not complete. We will extend and refine it with specific non-functional requirements, part of Section 3.2.

Based on the requirements identified before, we derived the following components: instrumentation code, agents, server, visualization and relevance feedback. Figure 3.1 representing the high level architecture shows how all these components fit together. In the rest of this section we briefly introduce each of the previously mentioned components.

- *instrumentation code* - is the code that is inserted into the application that has to be monitored. This code performs measurements of various metrics for the application being instrumented, in our case LarKC plug-ins, work-flow or LarKC platform and its components.

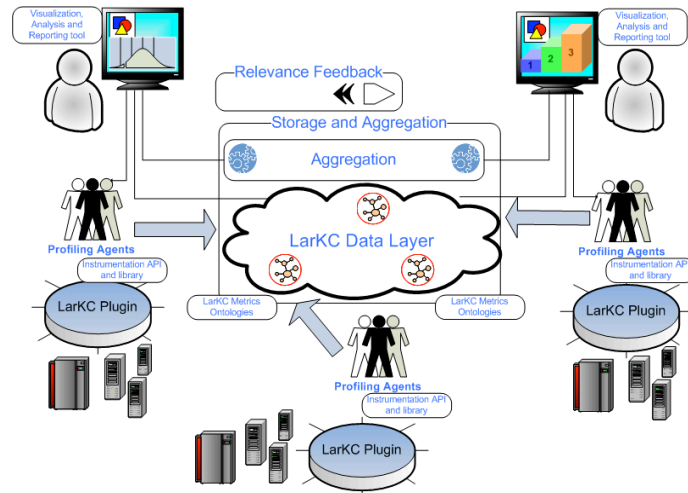


Figure 3.1: Instrumentation and monitoring architecture

- *agents* - are software artifacts that are deployed close to the resources being monitored. They are responsible for collecting and buffering the measurement data that is provided by the instrumentation code. The agents also communicate with the server, the central point where the data is stored. Agents also perform system measurements for the entire system (node) on which the various applications that are instrumented are running.
- *server* - is one of the core components that is responsible with the storage and aggregation of the monitored data, which is received from the profiling agents. Furthermore, the server provides the means to access, retrieve and query the measurement data.
- *visualization* - enables easy interpretation of the data collected during instrumentation and monitoring. It provides the means to visualize real-time as well as historical data, to display useful reports for the user of instrumentation and monitoring.
- *relevance feedback* uses data mining techniques and performs machine learning on top of the instrumented data. Its major role will be to experiment machine learning algorithms that based on instrumented data, predict different metrics given a certain input to the platform.

## 3.2 System components requirements

This section provides a more detailed description of functional and non-functional requirements of each system component.

### 3.2.1 Instrumentation requirements

In a nutshell, instrumentation can be seen as introducing new code in key parts of the application that, once executed, performs measurements which then are made available to the monitoring system. By consequence the general functional requirement of instrumentation is to provide code that enables measurements of various metrics.



This very general functional requirement can be refined into the following more specific functional requirements.

### Functional requirements

- The instrumentation should support different granularity of manipulation. More precisely from the general perspective, instrumentation should be done at the level of method, JVM, system and application. Using and extending this general perspective, specific instrumentation will be performed for the LarKC plug-ins and platform components. In the case of the LarKC project, the instrumentation must be at different levels i.e. plug-in, work-flow, platform.
- The instrumentation should make measurements of various metrics that are relevant for the person monitoring the system. The metrics of interest and the methods to be monitored will be identified via various mechanisms e.g. annotations, API, library, etc.
- The instrumentation should make the measurements available to other components of the monitoring platform either through a pull or push approach.

### Non-functional requirements

It is important to stress that by introducing new code that performs measurements and executing this code instrumentation introduces overhead. By consequence, an important goal for instrumentation is to have a low overhead, such that it has no real performance impact on the instrumented application runtime. *A low overhead, low impact* is one of the key non-functional requirement for the instrumentation.

The overhead introduced by instrumentation can manifest itself at:

- developing time (adding the instrumentation to the application)
- compile time (the application takes longer to compile)
- runtime: class load time (because code injection is done when the JVM loads classes) and execution time (the extra time taken by instrumentation to do the actual measurements)

Instrumentation can be added to an application either by direct source code editing or by changing the already compiled sources (byte-code manipulation). There is also a mixed mode in which source code is marked, using Java annotations for example, to be instrumented by byte-code manipulation. In case of byte-code instrumentation, it can be done at compile time or at runtime.

Another key non-functional requirement for the instrumentation is the ability to gather *fast*, as much useful information as possible, but at the same time to keep the low overhead. In this context, one must be able to answer three questions: *where?*, *when?*, *what?*, meaning *where* to inject the instrumentation code, *when* should the instrumentation code be executed and *what* metrics should the instrumentation code measure.

To summarize, we identified the following non-functional requirements for the instrumentation.



- The instrumentation should introduce a low, minimal measurement overhead. Also known as, non-intrusiveness, this non-functional requirement is specific not only to instrumentation but to all the other components of the monitoring solution. In short we want the instrumentation to minimize as much as possible the usage of computation resources (CPU, memory, etc.). This can be generalized to the usage of other type of resources, for example network bandwidth for communication.
- The instrumentation should be characterized by low latency. More precisely instrumentation should gather measurements and pass them as fast as possible to the upper levels of the monitoring platform.
- The instrumentation should be easy to use and should introduce a minimal overhead for the developer of the business logic code. More precisely in the case of LarKC, instrumenting the code should be made transparent to the LarKC developer as much as possible. For example the instrumentation should be inserted into the application being monitored preferably at run time, thus being transparent to the developer.
- The instrumentation should be portable. Given that the LarKC plug-ins and components can run on a wide spectrum of hardware and operating systems, instrumentation should be multi-platform, multi-operating system.
- There should be a way to turn off or on the instrumentation mechanism.

### 3.2.2 Agent requirements

In this section we describe the functional and non-functional requirements that the agent component of the instrumentation and monitoring solution should address.

#### **Functional requirements**

We identified the following functional requirements for the agent component.

- The agents should be customizable with respect to the metrics they gather information about. Furthermore, if new metrics are added or existing metrics are changed, the agent should cope easily with these changes.
- The agent should collect and buffer measurements data received from the instrumentation code.
- The agent should send the measurements to the server
- The agent should be deployed next to the instrumentation code.
- The agent should perform measurements of metrics that are relevant for the entire node where the agent is deployed i.e. the computer where the instrumentation code and agent are running
- The agent should be reliable when it comes to measurements that it collects and sends to the server. For example if by any reason the agent terminates but it has already collected some measurements that were not sent to the server, when restarted, the agent must be able to resend the rest of the data to server.



- The agent should be able to auto-discover the server if deployed on the same network.

### **Non-functional requirements**

We identified the following non-functional requirements for the agent component.

- The agent should be scalable. It should be prepared to gather considerable amounts of data in real time from the multiple instances of instrumentation code.
- The agent should be fault tolerant, meaning that if the data that it receives from the instrumentation code is corrupted the agent should carry on its functionality without interruption.
- The agent should communicate asynchronously with the instrumentation code and with the server.
- The agent should be non-blocking.
- The agent should be self-configurable.
- The agent should be portable. Same as the instrumentation code, the agent should run on a wide spectrum of hardware and operating systems.

#### **3.2.3 Server requirements**

The server is a core component of the overall infrastructure and is responsible for collecting and aggregating measurement data from the agents and making the data available to the relevance feedback and visualization component.

### **Functional requirements**

We derived the following functional requirements that the server component should fulfill:

- The server should receive measurement data from the agents and should store this data in a persistent way.
- The server should provide means to access and retrieve measurement data.
- The server should provide means to query the measurement data that it stores.
- The server should expose measurement data as semantic data and if necessary transform other data formats (e.g. Java model) to semantic model.
- The server should create compound metric measurements given atomic metric measurements received from the agents.



## Non-functional requirements

We derived the following non-functional requirements that the server component should fulfill:

- The server should be scalable, because there are potentially thousands of agents deployed in the distributed environment that is monitored.
- The server should be fault tolerant. It should tolerate possible failures of nodes, agents and components.
- The server should communicate asynchronously with the agents.
- The server should be non-blocking.
- The server should be portable. Similarly to the instrumentation code and the agent, the server should run on wide spectrum of hardware and operating systems.

### 3.2.4 Visualization requirements

Visualization should be done via an interactive component providing a two way communication channel: first it gets the stored and aggregated data and provides a nice visual view of it and secondly it allows users select what they want to visualize from the available atomic or compound metrics, it provides users the possibility to input some query or work-flow configurations that will be send to the relevance feedback mechanism which, in its turn, will communicate to the visualization tool some data (in the form of metrics, or plug-in names, or work-flow parameters).

## Functional Requirements

The Visualization Component behavior is described by the following functional requirements:

- The visualization component should collect some input from the users. (For example the user may adjust interactively visualization parameters or the user may customize his dashboard with the widgets according to his needs).
- The visualization component should interact with the data storage and aggregation.
- The visualization component should interact with the relevance feedback module.
- The visualization component should be able to provide a more intuitive visual representation of the stored data.

## Non-functional Requirements

- Integration:
  - The visualization functions and data should be grouped in widgets. Many widgets should be grouped in categories based on the visualization purposes and user preferences.



- Access Control:
  - The user must login in order to have access to the visualization widgets.
- Easily editable content:
  - The content provided by the visualization module should be easily editable.
- Extensibility:
  - The visualization component modules should be easily installed and extended.
- Back-up/Restore:
  - The system should be able to back-up and restore data.
- Response Time:
  - The visualization component should be characterized by a good response time.
- Availability:
  - The visualization platform should be available for use at any time.
- Portability:
  - The visualization component should run on main operating systems (Windows, Linux, etc.).
- Usability:
  - The visualization component should be simple, easy to use and understand.

### 3.2.5 Relevance feedback requirements

The positioning of the relevance feedback component within the whole instrumentation and monitoring architecture is depicted in figure 3.2. We can notice that the relevance feedback component has a tight interaction with the storage and aggregation component and with the visualization component. Hence, the requirements will have to consider these interactions.

#### **Functional requirements**

The specific behavior of the relevance feedback component is indicated by a set of functional requirements:

- Robust interaction with the data storage and aggregation component
- Robust interaction and integration within the visualization component
- Accurate prediction: refers to the ability of the relevance feedback component to correctly predict the performance metrics on new or previously unseen data. Similarly, the accuracy of a predictor refers to how well a given predictor can guess the value of the predicted attribute for new or previously unseen data.

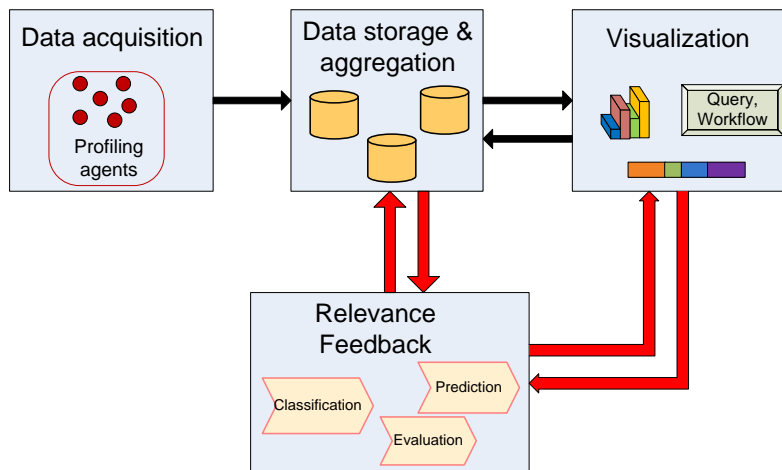


Figure 3.2: Relevance feedback component within the monitoring and instrumentation architecture

- Accurate feature analysis and selection
- Accurate data clustering

### Non-functional requirements

For describing the overall operation of the relevance feedback component we have identified a set of non-functional requirements that should ensure a good execution. The list of non-functional requirements is:

- Speed refers to the computational costs involved in generating and using the classification or prediction functionalities provided by the relevance feedback component. The relevance feedback should be able to provide a prediction response as fast as possible.
- Robustness represents the ability of the relevance feedback component to make correct and accurate predictions given noisy data or data with missing values.
- Portability indicates that the relevance feedback component can be used in any environment (Linux, Windows, other) in which the visualization component is running.



## 4. State of the Art

In this chapter a survey and analysis of tools for instrumentation, monitoring solutions including agents and server, and visualization is performed.

### 4.1 Instrumentation

In this section we investigate a set of tools that can be used for code instrumentation (Subsection 4.1.1) and also tools for instrumentation and monitoring of Java applications (Subsection 4.1.2).

#### 4.1.1 Java tools that can be used for code instrumentation

**ASM**<sup>1</sup> is the de-facto low level Java byte-code manipulation tool. It is focused on simplicity of use and performance, it's small, fast and very robust, having a good community and good documentation. Because it was designed and implemented to be as small and as fast as possible, it makes it very attractive for using in dynamic systems (can of course be used in a static way too). The project uses a FLOSS license, the ASM license<sup>2</sup>, which is very similar with a BSD license. Working with ASM is very similar with SAX or DOM, in the sense that it has two APIs for manipulating Java byte-code: event based or tree/object based. The event based API works by taking as input a compiled Java class and then triggering events for each 'interesting' element from the class and allowing the programmer to specify a transformation on that element. With the object based model, a class is represented with a tree of objects, each object representing a part of the class, such as the class itself, a field, a method, an instruction, etc. Because ASM deals directly with byte-code it also imposes on the developer to be familiar with Java byte-code (Java byte-code is very close to assembly code) and JVM internals.

**BTrace**<sup>3</sup> is a safe, dynamic tracing tool for Java. BTrace works by dynamically (byte-code) instrumenting classes of a running Java program. BTrace inserts tracing actions into the classes of a running Java program and hot swaps the traced program classes. It uses ASM to do the actual byte-code manipulation. The project uses a FLOSS license, GPL v2 with Classpath Exception<sup>4</sup>. Working with BTrace means understanding three concepts: probe points, actions and action methods. Probe points define "location" or "event" at which a set of tracing statements are executed. Probe point is "place" or "event" of interest where we want to execute some tracing statements. Actions define trace statements that are executed whenever a probe "fires". Action methods are Java methods from a class that encapsulate the trace statements that are executed when a probe fires (actions are defined inside action methods). To guarantee that the tracing actions are "read-only" (i.e., the trace actions don't change the state of the program traced) and bounded (i.e., trace actions terminate in bounded time), a BTrace program is allowed to do only a restricted set of actions. A short example on how to use BTrace for instrumentation is available in listing 4.1.

---

<sup>1</sup><http://asm.ow2.org>

<sup>2</sup><http://asm.ow2.org/license.html>

<sup>3</sup><http://kenai.com/projects/btrace>

<sup>4</sup><https://visualvm.dev.java.net/legal/gplv2+ce.html>



Listing 4.1: BTrace example

```
@BTrace
class Instrumentation {
    @OnMethod( clazz="com.company.app.ClassToInstrument",
              method="methodToInstrument")
    void instrumentationCode() { // instrumentation code
        here }
}
```

Listing 4.2: JBoss AOP example

```
@Aspect
public class Instrumentation {
    @Bind (pointcut="execution(* com.company.app.
            ClassToInstrument->methodToInstrument(..))")
    public Object methodAdvice(MethodInvocation invocation
        ) throws Throwable { // instrumentation code here }
}
```

**JBoss AOP**<sup>5</sup> is a Java aspect oriented framework implementation. Aspect-oriented programming is a way of modularizing crosscutting concerns much like object-oriented programming is a way of modularizing common concerns. It uses javassist<sup>6</sup> to do the actual byte-code manipulation. The project uses a FLOSS license, LGPL<sup>7</sup>. Working with JBoss AOP requires understanding the AOP concepts: join-point, point-cut, advice, aspect. Join-point is a well-defined point in the program flow (method call/execution, constructor call/execution, field access, etc.). A point-cut picks out certain join points and values/context at those points (or put another way: as a regular expression matches strings, a point-cut expression matches a particular join-point or a list of join-points). A short example on how to use JBoss AOP for instrumentation is available in listing 4.2.

**AspectJ**<sup>8</sup> is a Java aspect oriented framework implementation. Aspect-oriented programming is a way of modularizing crosscutting concerns much like object-oriented programming is a way of modularizing common concerns. It uses ASM to do the actual byte-code manipulation. It also integrates very close with Eclipse with the AJDT plug-in. It allows the byte-code manipulations to be done either at compile time (with ajc) or runtime (weaving). AspectJ introduces the notion of join-point, that is just a name for an existing Java concept. Overall, AspectJ it adds to Java only a few new constructs: point-cuts, advice, inter-type declarations and aspects. Point-cuts and advice dynamically affect program flow, inter-type declarations statically affects a program's class hierarchy, and aspects encapsulate these new constructs. The project uses a FLOSS license, EPL v1<sup>9</sup>. Working with AspectJ can be done either using the aspect language or by using annotations on standard Java code. Combining the

<sup>5</sup><http://www.jboss.org/jbossaop>

<sup>6</sup><http://www.jboss.org/javassist>

<sup>7</sup><http://www.gnu.org/copyleft/lesser.html>

<sup>8</sup><http://www.eclipse.org/aspectj>

<sup>9</sup><http://eclipse.org/legal/epl-v10.html>



Listing 4.3: AspectJ example

```
aspect Instrumentation {
    pointcut methodAdvice(): execution(* com.company.app.
        ClassToInstrument.methodToInstrument(..));
    before(): methodAdvice() { // instrumentation code
        here }
}
```

Listing 4.4: Perf4J example

```
class ClassToInstrument {
    @Profiled
    void methodToInstrument() { // code for the actual
        method}
}
```

aspect language with AJDT plug-in for Eclipse allows the programmer to see the code affected by aspects, to benefit from AspectJ-aware content assist and eager parsing. As with JBoss AOP the working with AspectJ requires understanding the AOP style of programming. A short example on how to use AspectJ for instrumentation is available in listing 4.3.

#### 4.1.2 Java instrumentation and monitoring tools

**Perf4J**<sup>10</sup> is a set of utilities for calculating and displaying performance statistics for Java code. For the moment it only supports wall clock time (min, max, avg) and invocation count metrics (count, tps). It provides source code and mixed mode instrumentation. In case of mixed mode instrumentation it uses AspectJ for the bytecode manipulation. The project uses a FLOSS license, Apache v2<sup>11</sup>. Working with Perf4j can be done by either using the provided API or by using annotations. In the first case for each method or block of code that we want to instrument we need to instantiate a StopWatch at the beginning; and call stop at the end. In case of using annotations, it works only on methods (not on arbitrary code blocks) and all is needed is annotating the method with @Profiled annotation. When the application runs measurements are taken on the instrumented parts and this measurements are written to a log like file. Perf4j offers then an reporting application to aggregate, build graphs, publish to JMX the collected measurements from the log files. A short example on how to use AspectJ for instrumentation is available in listing 4.4.

**Parfait**<sup>12</sup> is a performance management and monitoring framework for Java 1.6. parfait consists of several modules: a core monitoring subsystem, which defines a series of monitorable values and counters (which can be of any Java type) and some basic operations on these, along with standard interfaces for exporting these values to various 'data sinks' as they change over time. The parfait project defines a few common data sinks for monitoring the current state of these values, exporting them to JMX and

<sup>10</sup><http://perf4j.codehaus.org>

<sup>11</sup><http://www.apache.org/licenses/LICENSE-2.0>

<sup>12</sup><http://code.google.com/p/parfait>



Listing 4.5: Parfait example

```
class ClassToInstrument {
    MonitoredCounter counter = new MonitoredCounter("
        methodToInstrument.executions", "");
    MonitoredCounter success = new MonitoredCounter("
        methodToInstrument.success", "");
    MonitoredCounter fail = new MonitoredCounter("
        methodToInstrument.success", "");

    void methodToInstrument() throws Exception {
        counter.inc()
        try {
            // real method code here
            success.inc();
        } catch (Exception e) {
            fail.inc();
            throw e;
        }
    }
}
```

SGI's PCP monitoring system. There are also a number of modules built on top of this core system, which enable the collection of metrics from common data sources (such as JDBC drivers and other JMX beans) to provide easy monitoring of existing Java subsystems. It provides source code instrumentation and for Spring applications also provides byte-code instrumentation by using String DI and AspectJ. The project uses a FLOSS license, Apache v2<sup>13</sup>. Working with Parfait can be done by using the provided API. An important concept defined by Parfait is `Monitorable`. This encapsulates all the key metadata for an observable monitored value, and can be used by the various output subsystems to display, log, or otherwise use the value in an appropriate way. Different types of `Monitorable` are defined: `MonitoredCounter` (which covers values with increasing-counter semantics), `MonitoredValue` (which is used for free-running point-in-time values which can be set arbitrarily), `PollingMonitoredValue` (which uses a `Timer` and a callback mechanism to poll for `Monitorable` values from sources which do not provide asynchronous notifications). All the data gathered by `Monitorables` can be sent to different output mechanisms. A particular parfait output library will typically implement its own `MonitoringView`, which is responsible for initializing the monitoring output and being notified (via a `Monitor`) of any changes to the `Monitorables` it obtains from the provided `MonitorableRegistry`. A short example on how to use AspectJ for instrumentation is available in Listing 4.5.

**Usemon**<sup>14</sup> is a monitoring system for usage trends, response time and dependency analysis of plain Java applications or big multi-clustered Java Enterprise applications running in production. It uses `javassist` to do the actual byte-code manipulation. It offers little extensibility or customization for code instrumentation. The only customization allowed is to provide the names of the custom classes that need to be

<sup>13</sup><http://www.apache.org/licenses/LICENSE-2.0>

<sup>14</sup><http://code.google.com/p/usemon>



instrumented. The project uses a FLOSS license, MPL v1.1<sup>15</sup>. The Usemon system consists of three separate components: agent, collector, visualizer/ui. The agent performs byte-code manipulation of interesting classes, injecting statistics gathering code in all public methods of classes like EJBs, MDBs, SQL statements, Queue senders or configured POJOs for that matter. Data is collected in intervals of 60 seconds and transmitted to the collector for storage. The Usemon Collector receives the data from the Usemon Agents, transforms the observations and stores them into the SQL database. The data model of the database is based upon a star scheme and allow very fast slice and dicing of the information. There are currently two UIs made for Usemon. The first is a GWT based web application and the other one is Usemon Live, a stand alone Java application that is meant for big screens and for monitoring.

## 4.2 Monitoring approaches - agents and server

Monitoring distributed applications is a large, widely investigated area. Many approaches exist, both from academia and especially from the industry. In this section we have a look at some of the exiting approaches for monitoring. We describe briefly how each approach works and what are its main characteristics. A special attention is given to how data is collected i.e. the monitoring agents, and how data is stored, aggregated, reported, etc., i.e. the server side. Other aspects, such as visualization, supported metrics, type of licenses are also shortly investigated.

**Hyperic HQ**<sup>16</sup> is a monitoring solution for Web applications that can be used for a large set of platforms, servers and technologies. The Hyperic is an agent based solution that supports the collection of measurements for a large set of metrics using HQ agents, the discovery of resources and information about these resources is done as well by HQ agents, the storing of information about resources in a HQ hierarchical inventory model. Other tasks supported by Hyperic HQ is the ability to remotely control and administrate software resources. Available control actions vary by resource type, for example one can start, stop software resources as well as perform analysis or housekeeping functions. Hyperic HQ includes as well an alerting mechanism. When an alert fires, HQ can respond in a variety of ways: it can issue email notifications, set SNMP traps, perform a control action, or issue a communication to another management system. Last but not least, Hyperic HQ has a highly configurable user interface in a form of a dashboard, for monitoring and analyzing performance and availability. Hyperic HQ is extensible i.e. on the server side one can develop plug-ins that manage resources for which HQ does not provide support and on the visualization part, the dashboard can be extended with customized views. Hyperic HQ can be integrated with other monitoring solutions such as Nagios and OpenNMS, that are described as well in this section.

- *Data Storage*  
Hyperic HQ storage module relies on relational databases. MySQL, Oracle and Postgresql are some of the database servers that can be integrated with Hyperic.
- *Reporting*  
An integrated part of the Hyperic HQ solution is the reporting module. Hyperic

---

<sup>15</sup><http://www.mozilla.org/MPL>

<sup>16</sup><http://www.hyperic.com/>



HQ creates reports on the loading, capacity and resource usage. Furthermore, Hyperic HQ reporting module can generate an analysis of available resources, alerts and history of events of the monitored system. Hyperic HQ allows as well the definition of custom reports. All reports can then be exported in different formats such as Pdf, HTML, Excel and CSV. A nice feature of Hyperic, called “metric baselines”, refers to automatic computation of future values of a metric given previous values. One can configure alerts in Hyperic HQ relative to “metric baselines”.

- *Viewing results*

Hyperic HQ provides Web-based visualization of the monitored data through a dashboard. Using the dashboard the user can get an overview of the resources monitored, recent alerts, graphics of various metrics, etc. The dashboard is very configurable, being made of portlets that can be easily added, removed, re-arranged and configure. The dashboard offers as well the possibility to manage the monitoring system and the alerts. In the enterprise version of Hyperic HQ, the user can also save the evolution graphics of various metrics and can visualize alerts for groups of resources.

- *Metrics*

Hyperic HQ supports a large set of generic metrics, such as availability, performance, utilization, and throughput. The set of standard metrics is extensible with other custom metrics that can be defined through the dashboard. The metrics of interest can be selected using the dashboard as well.

- *License*

Hyperic HQ is distributed under the GPL General Public License version 2.

- *Java and JMX support*

Hyperic HQ was developed using Java technologies and has support JMX.

**OpenNMS**<sup>17</sup> is a distributed, scalable platform for monitoring. OpenNMS gathers data for monitoring from nodes via protocols including SNMP, JMX, HTTP, Windows Management Instrumentation, and NSClient. OpenNMS can also be used for determining service availability and latency, including distributed measurement of availability and latency, and reporting of the results. Event management is another functionality that is implemented in OpenNMS. More precisely, OpenNMS receiving events, both internal and external, including via SNMP traps. OpenNMS also includes alarm handling. It reduces events according to a reduction key and scripting automated actions centered around alarms. Notification is also available in OpenNMS. More precisely, OpenNMS can send notices regarding noteworthy events via e-mail, XMPP, or other means.

- *Data Storage*

The data collected by the OpenNMS monitoring system is stored in Round Robin Database (RRD) format, in JRobin<sup>18</sup> files.

- *Reporting*

The RRD files can be used to generate graphical reports, including reports containing data collected from various devices, protocols and sources (SNMP, ICMP,

---

<sup>17</sup><http://www.hyperic.com/>

<sup>18</sup><http://oldwww.jrobin.org/>



HTTP), predefined reports on various metrics, SLA reports on availability for example, resource graphs of performance and latency, charting reports, etc. Reports can be exported in different formats such as Pdf, HTML, XML, and SVG. OpenNMS can also export performance metrics to a remote service using TCP.

- *Viewing results*

Visualization of OpenNMS monitored data is done using the JRobin Inspector<sup>19</sup>. JRobin is a tool for visualization for RoundRobinDatabase (RRD) files.

- *Metrics*

Some of the metrics that OpenNMS is aware of are temperature, CPU load, disk usage, availability, performance, etc. OpenNMS also implements a distributed monitoring for most protocols (HTTP, HTTPS, SNMP, LDAP, ICMP, FTP, etc.)

- *License*

OpenNMS is distributed under the GPL General Public License.

- *Java and JMX support*

OpenNMS was developed in Java. It supports different implementation of JMX agents (boss, mx4j and JSR 160) OpenNMS interface was developed using GoogleWebToolkit (GWT)<sup>20</sup>.

**Nagios**<sup>21</sup> is a system and network monitoring application that provides monitoring of all mission-critical infrastructure components – including applications, services, operating systems, network protocols, system metrics, and network infrastructure. Nagios does not include any internal mechanisms for checking the status of hosts and services. Nagios relies on external programs (called plug-ins) that actually do the work. These are executable scripts that can check some status, host or service. Once a plug-in is executed, it returns the result to Nagios which will process the results that it receives from the plug-in and take any necessary actions (running event handlers, sending out notifications, etc). At a minimum, Nagios plug-ins must return a single line of human-readable text that indicates the status of some type of measurable data. Nagios sends notifications when service or host problems occur and get resolved (via email, pager, or user-defined method). It has the ability to define event handlers to be run during service or host events for proactive problem resolution. Nagios also provides the ability to define network host hierarchy using “parent” hosts, allowing detection of and distinction between hosts that are down and those that are unreachable. Furthermore, Nagios has an extensible architecture. Integration with in-house and third-party applications is easy, with multiple APIs. Hundreds of community-developed add-ons extend core Nagios functionality. It has a simple plug-in design that allows users to easily develop their own service checks.

- *Data Storage*

Nagios stores data in text files in the log directory, unless Nagios database extensions are used. Nagios provides a set of add-ons for storing data in a database. For example the NDOUtils add-on allows storing all status information from Nagios in a MySQL database. Multiple instances of Nagios can all store their information in a central database for centralized reporting.

---

<sup>19</sup><http://oldwww.jrobin.org/>

<sup>20</sup><http://code.google.com/webtoolkit/>

<sup>21</sup>[www.nagios.org/](http://www.nagios.org/)



- *Reporting*  
Nagios comes with a dedicated tool for reporting, called NaReTo. NaReTo stand for Nagios Reporting Tools and give administrators reporting and availability of servers and groups. It provides the means to manage SLA Reports and provides historical records of outages, notifications, and alert response for later analysis. Third-party add-ons are available to extend reporting capabilities. Nagios Web interface is a multi-user access (role based). User-specific views ensure clients see only their infrastructure components. Based on information from Nagios, NaReTo provides views of the highest level in different categories of users. Three views are available: the Real Time View, Reporting, the History Alarms and Monitoring Alarms.
- *Viewing results*  
Nagios provides a centralized view of entire monitored IT infrastructure. Detailed status information is available through web interface: current network status, notification, problem history, log file, etc. Nagios Web interface is a multi-user access (role based). User-specific views ensure clients see only their infrastructure components.
- *Metrics*  
Nagios can monitor many different types of services, protocols and devices: various protocols (e.g. HTTP, POP3, IMAP, FTP, SSH, DHCP), CPU Load, Disk Usage, Memory Usage, Current Users, Unix/Linux, Windows, and Netware Servers Routers and Switches, network printers etc.
- *License*  
Nagios is distributed under the GPL General Public License version 2.
- *Java and JMX support*  
JMX support is done via the plug-in mechanism. Nagios is not implemented in Java, but there exist plug-ins for monitoring Java applications and application servers.

**Zabbix**<sup>22</sup> is a complete, open source monitoring solution that can be used to monitor numerous parameters of a network and the health and integrity of servers. Zabbix offers a flexible notification mechanism that allows users to configure e-mail based alerts for virtually any event. Reporting and data visualization are an integrated part of the Zabbix solution. The Zabbix architecture includes the following components: server, proxy, agent and interface. The Zabbix server can remotely check networked services. It is a central component to which Zabbix agents will report availability and integrity information and statistics. The server can be seen as a central repository in which all configuration, statistical and operational data are stored. The Zabbix proxy is an optional part of Zabbix that collects performance and availability data on behalf of Zabbix Server. Monitoring of the local resources and applications is done by the Zabbix agent that uses native system calls for gathering statistical information for the system on which it is running. The Zabbix interface is a part of the server and provides reporting and visualization of the monitored data. Zabbix also offers an API for programmable interface to Zabbix deployment.

---

<sup>22</sup><http://www.zabbix.com/>



- *Data Storage*  
Zabbix keeps history of all data and measurements and saves it in a relational database. For storage support, Zabbix can be integrated with various database servers such as MySQL, Oracle, PostgreSQL and SQLite. Data can be exported as XML, which makes it easy to read and modify. The use of XML format makes possible integration and data import/export with third party tools and applications.
- *Reporting*  
Zabbix supports easy integration of external tools for reporting. Zabbix offers customized statistics for different time intervals (e.g. yearly/monthly/daily) and provides as well real-time Service Level Agreement reports.
- *Viewing results*  
The visualization part of Zabbix is web based, and Zabbix results can be visualized using standard web browsers. An integrated part of Zabbix is the Zabbix Dashboard that provides high level personalized details about the monitored environment. Maps, screens and graphs can be accessed from the Dashboard. Zabbix screens allow grouping of various information for quick access and display on one screen. Zabbix screens can be easily created and can visualize information in various ways such as simple graphs, maps, user-defined graphs, plain text, tables, charts (both 2D and 3D), etc. Zabbix also provides a historical perspective of the monitored data, including history of events and actions.
- *Metrics*  
Zabbix can be used to monitor various metrics including performance related metrics, availability, etc. all of them for Web applications. Monitoring of low level metrics such as memory and network utilization, disk I/O, disk space availability, file checksums, and monitoring of log files are also possible with Zabbix. Among the protocols, services and devices that can be monitored by Zabbix we mention HTTP, HTTPS, SNMP, IPMI devices, FTP and SSH.
- *License*  
Zabbix is written and distributed under the GPL General Public License version 2.
- *Java and JMX support*  
Integration with JMX is possible by defining user parameters for the Zabbix agent in its configuration file.

## 4.3 Data visualization

### 4.3.1 Introduction

The visualization, analysis and reporting tool allows us to enable easy interpretation of the data collected during instrumentation and monitoring. Using this tool, LarKC users and developers can visualize real-time or historical data. Furthermore they can analyze reports based on aggregated metric values. An important requirement for the data visualization tool is to cope with a large amount of data. The need for interactive data analysis and reporting is becoming even more important when we deal with incomplete growing data sets. A challenging task is to extract, analyze, and use



relevant knowledge included in such a mix of data. The visualization components offer a possibility to interactively extract and identify important patterns in a large amount of information. Further, visualization results allow users to discover the potential anomalies and make appropriate decisions. In developing a visualization system the following aspects should be taken into account: the visualization domain (features of interest) and the way users perceive and interact with visualization results [3]. For Instrumentation and Monitoring the features of interest are given by the atomic and compound metrics at the Platform, Work-flow and Plug-in levels.

Nowadays we can distinguish between different areas of visualization [4]:

- **Information visualization:** is usually applied to the visual representation of large-scale collections of non-numerical information, such as bibliographic databases, files, lines of code in software [5], ontology models.
- **Scientific visualization:** uses visualization of phenomena in architecture, meteorology, medicine, biology, etc.
- **Data visualization:** includes representation of data, which has been abstracted in some schematic form, including attributes or variables for the units of information.

There are different forms of visualization. The basic objective of all types of information visualization is to transform large data into a more intuitive visual representation such as relational graphs, charts, maps, pies, boxes, bars etc.

#### 4.3.2 Visualization architecture and components

Figure 4.1 shows a generic model with the main modules of a visualization system. We consider that the visualization data is integrated on a web application server. The reports are given to the user in a web browser through the HTTP requests and responses. The reports are encapsulated into widgets or portlets. User can customize his interface by visualizing a subset of custom widgets according to his needs. A controller of business logic coordinates requests from clients, as well as the data layer queries and responses. Based on client requests and data base (data layer) query results, actions are carried out by the server.

A visualization system may contain the following main components:

An **application server** represents a framework dedicated to the efficient developing, deploying, execution and integrating construction of applications. For web applications, the main job of an application server is to support the construction of dynamic pages. The visualization and reporting data model can be stored in a **database**. The connections to the database and to the **LarkC Data Layer** are handled by the application server. Depending on the technology used, the visualization interface can be developed by using and extending **enterprise portals** or **Content Management Systems**. The aim is to construct a scalable way to aggregate, personalize and report de-centralized content through application specific user interface containers, like **portlets** (specified in JSR168 and JSR 286 - Java Portlet Definition Standards) or **widgets**. Users can interactively view or customize their reporting interface. The client requests and responses are coordinated by the application server.

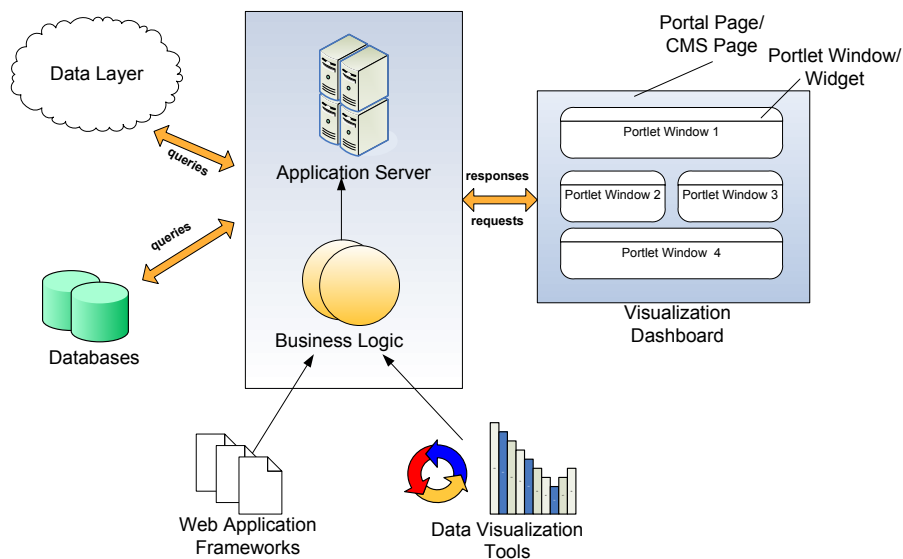


Figure 4.1: Data visualization and reporting

A **web application framework** represents a software framework designed to support the development of dynamic websites, web applications and web services, providing libraries for database access, templates, session management etc. Most web application frameworks are based on the Model View Controller pattern by separating the data model with business rules from user interface. The visualization results may be presented to the user in many ways. The multitude of data visualization can be achieved by using different **data visualization tools** able to transform data into a more intuitive representation such as charts, maps, graphs, boxes, bars, tables etc. Next we will provide a list of main used tools for web development, content management and data visualization.

### 4.3.3 Web development frameworks

#### PHP Based

- **CakePHP** is an open source development framework for PHP providing an extensible architecture for developing, maintaining, and deploying applications. Using commonly known design patterns like MVC and ORM within the convention over configuration paradigm, CakePHP reduces development costs and helps developers write less code.
- **CodeIgniter** is a powerful PHP framework with a very small footprint, built for PHP coders who need a simple and elegant toolkit to create full-featured web applications.



- **Drupal**<sup>23</sup> is a free and open source content management system written in PHP and distributed under the GNU General Public License. It is used as a back-end system for at least 1% of all websites worldwide<sup>24</sup>. Drupal core is designed to be modular with a system of hooks and callbacks, which are accessed internally via an API. This design allows third-party contributed (often abbreviated to “contrib”) modules and themes to extend or override Drupal’s default behaviors without changing Drupal core’s code. Drupal’s modular design, which isolates Drupal core’s files from contributed module and themes, increases flexibility and security and allows Drupal administrators to cleanly upgrade to new releases of Drupal core without potentially overwriting their site’s customizations. To maintain this separation, Drupal administrators are instructed to avoid altering Drupal core’s software.
- **Horde**<sup>25</sup> framework offers applications such as the Horde IMP email client, a group-ware package (calendar, notes, tasks, file manager), a wiki and a time and task tracking software. The applications are under various Open Source licenses, mostly under the GNU Public License. The Horde Framework itself, as of version 2.0, is released under the LGP.
- **Lithium**<sup>26</sup> is a full-stack framework that provides structure and conventions for rapid development. Some useful features are: integrated Unit Testing, Aspect Inspired Filter, Document Oriented Data Sources, Automatic output, Extensible plug-in, Static Models with Object Oriented.
- **Qcodo** development framework<sup>27</sup> is an open-source PHP web application framework which builds an Object Relational Model (ORM), CRUD (Create, Retrieve, Update, Delete) UI pages, and AJAX hooks from an existing data model. It additionally includes a tightly-integrated HTML and JavaScript form toolkit which interfaces directly with the generated entities. It is a robust, comprehensive framework which can be utilized by small and large Web applications alike.
- **Seagull**<sup>28</sup> is an OOP framework for building web, command line and GUI applications. Licensed under BSD, the project allows PHP developers to easily integrate and manage code resources, and build complex applications.
- **Symfony**<sup>29</sup> is a web application framework whose aim is to speed up the creation and maintenance of web applications and to replace repetitive coding tasks. It requires a few prerequisites for installation: Unix, Linux, Mac OS or Microsoft Windows with a web server and PHP 5 installed. It is currently compatible with the following Object-relational mappings: Propel and Doctrine.
- **Yii**<sup>30</sup> is a free, open-source Web application development framework written in PHP5 that promotes clean, DRY design and encourages rapid development. It comes packaged with tools to help test and debug your application, and has clear and comprehensive documentation.

---

<sup>23</sup><http://www.drupal.org>

<sup>24</sup>[http://w3techs.com/technologies/overview/content\\_management/all](http://w3techs.com/technologies/overview/content_management/all)

<sup>25</sup><http://www.horde.org/about/>

<sup>26</sup><http://lithify.me/>

<sup>27</sup><http://www.qcodo.com>

<sup>28</sup><http://seagullproject.org/>

<sup>29</sup><http://www.symfony-project.org/>

<sup>30</sup><http://www.yiiframework.com/about/>



- **Zend Framework**<sup>31</sup> is a use-at-will framework. There is no single development paradigm or pattern that all Zend Framework users must follow, although ZF does provide components for the MVC, Table Data Gateway, and Row Data Gateway design patterns. Zend Framework provides individual components for many other common requirements in web application development.

## Java Based

- **Apache Click**<sup>32</sup> is a JEE web application framework, providing rich client style programming model. Apache Click is designed to be easy to learn and use. It is a free and open-source project distributed under the Apache license and runs on any JDK installation.
- **Apache Cocoon**<sup>33</sup> is a Spring-based (since version 2.2 of Cocoon) framework built around the concepts of separation of concerns and component-based development. Cocoon implements these concepts around the notion of component pipelines, each component on the pipeline specializing on a particular operation. This makes it possible to use a Lego(tm)-like approach in building web solutions, hooking together components into pipelines, often without any required programming. The flexibility afforded by relying heavily on XML allows rapid content publishing in a variety of formats including HTML, PDF, and WML. The content management systems Apache Lenya and Daisy have been created on top of the framework. Cocoon is also commonly used as a data warehousing ETL tool or as middle-ware for transporting data between systems.
- **Apache Struts**<sup>34</sup> is a free open-source framework for creating Java web applications. Web applications differ from conventional websites in that web applications can create a dynamic response. Many websites deliver only static pages. A web application can interact with databases and business logic engines to customize a response. The framework provides three key components:
  - A “request” handler provided by the application developer that is mapped to a standard URI.
  - A “response” handler that transfers control to another resource which completes the response.
  - A tag library that helps developers create interactive form-based applications with server pages.
- **Apache Wicket**<sup>35</sup> is a lightweight component-based web application framework for the Java programming language conceptually similar to JavaServer Faces and Tapestry.
- **AppFuse**<sup>36</sup> is an open source project and application that uses open source tools built on the Java platform to help you develop Web applications quickly and efficiently. It was originally developed to eliminate the ramp-up time found

---

<sup>31</sup><http://framework.zend.com/>

<sup>32</sup><http://click.apache.org/>

<sup>33</sup><http://cocoon.apache.org/>

<sup>34</sup><http://struts.apache.org/>

<sup>35</sup><http://wicket.apache.org/>

<sup>36</sup><http://appfuse.org/display/APF/Home3>



when building new web applications for customers. At its core, AppFuse is a project skeleton, similar to the one that's created by your IDE when you click through a wizard to create a new web project.

- **Aranea**<sup>37</sup> is an open source Java MVC web framework that provides a common object-oriented approach to building the web applications, reusing GUI logic and extending the framework. It comes with out-of-the-box support for nested flows and database-backed query browsing. Additionally it serves as an integration platform, allowing free intermingling of arbitrary frameworks, components and applications.
- **Eclipse RAP**<sup>38</sup> (Rich AJAX Platform) enables software developers to build AJAX-enabled rich Internet applications by using the Eclipse development model, plug-ins and a Java-only API. It can be considered a counterpart for web development to the Rich Client Platform.
- **FormEngine**<sup>39</sup> is a Java Web Framework that enables simple definition and processing of complex and dynamic forms. The processing of a form involves the verification of the input data, calculation of the input based on the information from other input fields as well as dynamic activation or hiding of the data fields depending on the user input.
- **Google Web Toolkit**<sup>40</sup> (GWT) is a development toolkit for building and optimizing complex browser-based applications. Its goal is to enable productive development of high-performance web applications without the developer having to be an expert in browser quirks, XMLHttpRequest, and JavaScript. GWT is used by many products at Google, including Google Wave and the new version of AdWords. It's open source, completely free, and used by thousands of developers around the world.
- **Hamlets**<sup>41</sup> (previously known as IBM Servlet-based Content Creation Framework) is an open source system for generating dynamic web-pages developed by René Pawlitzek at IBM. A Hamlet is a servlet extension that reads XHTML template files containing presentation using SAX (the Simple API for XML) and dynamically adds content on the fly to those places in the template which are marked with special tags and IDs using a small set of callback functions. A template compiler can be used to accelerate Hamlets.
- **IceFaces**<sup>42</sup> is an integrated Ajax Java application framework that enables Java EE Ajax application developers to easily create and deploy thin-client rich Internet applications (RIA) in pure Java. ICEfaces is a fully featured product that enterprise developers can use to develop new or existing Java EE Ajax applications at no cost.
- **JavaServer Faces**<sup>43</sup> (JSF) is the standard component-oriented user interface (UI) framework for the Java EE platform. In terms which may sound more

<sup>37</sup><http://www.araneaframework.org/?pid=about>

<sup>38</sup><http://www.eclipse.org/rap/>

<sup>39</sup><http://www.form-engine.de/en/index.jsp>

<sup>40</sup><http://code.google.com/webtoolkit/>

<sup>41</sup><http://hamlets.sourceforge.net/>

<sup>42</sup><http://www.icefaces.org/main/home/>

<sup>43</sup><http://www.javaserverfaces.org/get-started>



familiar, it's a Java-based web framework. The fastest way to get started with a JSF project is to use a Maven archetype.

## Content management systems and portals

### PHP Based

- **Joomla**<sup>44</sup> is a free and open source content management system. It includes a model-view-controller framework that can also be used independently. Joomla uses object-oriented programming techniques and software design patterns, stores data in a MySQL database, and includes features such as page caching, RSS feeds, printable versions of pages, news flashes, blogs, polls, search, and support for language internationalization.
- **Papaya CMS**<sup>45</sup> is an open source content management system, free of charge and complying with open standards like XML as data format, XSLT as template language, and PHP for programming.
- **MODx**<sup>46</sup> helps even regular individuals manage content on their websites simply, quickly and intuitively. MODx is an Open Source PHP web application framework with a capable built-in Content Management System (CMS).
- **Tiki Wiki CMS Groupware**<sup>47</sup>, or simply Tiki, is a free and open source wiki-based, content management system written primarily in PHP and distributed under the GNU Lesser General Public License (LGPL) license. In addition Tiki contains a number of collaboration features allowing it to operate as a Geo-spatial Content Management System (GeoCMS) or Group-ware web application.
- **Wolf CMS**<sup>48</sup> is a content management system and is free software published under the GNU General Public License v3. Wolf CMS is written in the PHP programming language. Wolf CMS is a fork of Frog CMS.
- **Tribiq CMS**<sup>49</sup> is a UK-based consultancy specializing in custom web site design and development. Tribiq is the trading name for Tribal Limited (incorporated in England), and was previously called Tribal Internet. It is a browser-based web site content management system, and a platform for bespoke on-line applications. It is available in an open source community edition under the GNU General Public License. Other editions of Tribiq CMS are also available under a commercial license.
- **TYPO3**<sup>50</sup> is a free and open source content management system. It is released under the GNU General Public License. It can run on Apache or IIS on top of Linux, Microsoft Windows, OS/2 or Mac OS X.
- **Serendipity**<sup>51</sup> is a PHP based blog and web-based content management system. It is available under a BSD license. It supports PostgreSQL, MySQL, and SQLite

<sup>44</sup><http://www.joomla.org/>

<sup>45</sup><http://www.papaya-cms.com/community-home-page.983.en.html>

<sup>46</sup><http://modxcms.com/about/>

<sup>47</sup><http://info.tiki.org/tiki-index.php>

<sup>48</sup><http://www.wolfcms.org/>

<sup>49</sup><http://tribiq.com/>

<sup>50</sup><http://typo3.org/>

<sup>51</sup><http://www.s9y.org/>



database back-ends, the Smarty template engine, and a plug-in architecture for user contributed modifications.

- **Zikula**<sup>52</sup> is a free open source web application framework released under the GNU General Public License. It can be used to develop robust, secure, interactive and editable websites and web based applications. Zikula is written in object oriented PHP and is fully modular. It requires a database and may use leading platforms like MySQL, PostgreSQL, Oracle Database and Microsoft SQL Server.
- **CMS Made Simple**<sup>53</sup> is an open source package, built using PHP with support for MySQL and PostgreSQL. The template system is driven using the Smarty Template Engine. CMS Made Simple aims to provide easy development and customization with themes, add-on modules, dynamic menus, tags and translations.
- **Drupal**<sup>54</sup> core, contains basic features common to most CMSs. These include user account registration and maintenance, menu management, RSS-feeds, page layout customization, and system administration. The Drupal core installation can be used as a brochure ware website, a single or multi-user blog, an Internet forum, or a community website providing for user-generated content.
- **Ocportal**<sup>55</sup> is a free and open source content management system (CMS) written in PHP and based on a MySQL back-end database.
- **Concrete5**<sup>56</sup> is an open source content management system (CMS) for publishing content on the World Wide Web and intranets. It enables users to edit site content directly from the page. concrete5 allows users to edit images through an embedded editor on the page.
- **WordPress**<sup>57</sup> is an open source content management system (CMS), often used as a blog publishing application, powered by PHP and MySQL. It has many features including a plug-in architecture and a template system. Used by over 12% of the 1,000,000 biggest websites.

## Java Based CMS and Portals

- **jAPS 2.0**<sup>58</sup> (Java Agile Portal System) is an open source professional web platform that makes easier aggregation, publishing, access, customization and integration of information, services, processes and resources, compatible with accessibility international standards like WCAG 2.0, Italian Stanca Law, Section 508 and PAS 78, for both front end and back office.
- **OpenCms**<sup>59</sup> is an open source content management system written in Java. It is distributed by Framfab and Alkacon Software under the LGPL license.

<sup>52</sup><http://zikula.org/CMS/Zikula/>

<sup>53</sup><http://www.cmsmadesimple.org/about-link/>

<sup>54</sup>[www.drupal.org](http://www.drupal.org)

<sup>55</sup><http://ocportal.com/start.htm>

<sup>56</sup><http://www.concrete5.org/about/showcase/>

<sup>57</sup><http://en.wikipedia.org/wiki/WordPress>

<sup>58</sup><http://www.japsportal.org/jAPSPortal/>

<sup>59</sup><http://www.opencms.org/en/>



OpenCms requires a JSP Servlet container such as Apache Tomcat. It is a CMS application with a browser-based work environment, asset management, user management, work-flow management, a WYSIWYG editor, internationalization support, content version control, and more features.

- **DSpace**<sup>60</sup> is an open source software package that provides the tools for management of digital assets, and is commonly used as the basis for an institutional repository. It supports a wide variety of data, including books, theses, 3D digital scans of objects, photographs, film, video, research data sets and other forms of content. The data is arranged as community collections of items, which bundle bit-streams together.
- **dotCMS**<sup>61</sup> is an open source web content management system (wCMS) for building/managing websites, content and content driven web applications. dotCMS includes features such as support for virtual hosting, WebDav, structured content, clustering and can run on multiple databases PostgreSQL, MySQL, MSSQL and Oracle. It also includes standard wCMS features like page caching, template generation, and an API. There are a number of features and modules in dotCMS, including RSS feeds, AJAX calendar, a reporting engine, news listing, blogs, forums, user tracking and tagging.
- **Nuxeo EP**<sup>62</sup> is a comprehensive enterprise content management (ECM) platform. It has been designed to be robust, scalable and highly extensible, by using modern open source Java EE technologies, such as: the JCR, JSF, EJB3, JBoss Seam, OSGi, and a Service Oriented Approach. It can be used to develop both web-based server applications and Rich Client applications.
- **Magnolia**<sup>63</sup> is an open source content management system (CMS) developed by Magnolia International Ltd., based in Basel, Switzerland. It is based on JSR-170.
- **Hippo CMS**<sup>64</sup> is an open source information centered content management system. The Hippo CMS project was initiated and is maintained by Hippo. It's targeted at medium to large organizations managing content for multi-channel distribution like web sites and intra-nets. It facilitates an open and flexible way of using your information by following international accepted open standards.
- **Liferay Portal**<sup>65</sup> is an enterprise web platform for building business solutions. Liferay supports the following standards: iCalendar, JSR-168, JSR-127, JSR-170, JSR-286, JSF 2.0 and OpenSearch. It is built with the following technologies: j2EE, ehcache, Groovy, Hibernate, ICEfaces, jBPM, JGroups, Lucene, Seam, Spring 3.0, Struts, Tapestry and Velocity.
- **Jetspeed2**<sup>66</sup> is an open portal platform and enterprise information portal, written under the Apache license in Java and XML and based on open standards. Within a Jetspeed portal, individual portlets can be aggregated to create a page.

---

<sup>60</sup><http://www.dspace.org/>

<sup>61</sup><http://dotcms.com/>

<sup>62</sup><http://www.nuxeo.org/xwiki/bin/view/Main>

<sup>63</sup><http://www.magnolia-cms.com/home/>

<sup>64</sup><http://www.onehippo.org/>

<sup>65</sup><https://www.liferay.com/home>

<sup>66</sup><http://portals.apache.org/jetspeed-2/>



Each portlet is an independent application with Jetspeed acting as the central hub making information from multiple sources available in an easy to use manner. Jetspeed has been fully conforming to the Java Portlet 2.0 Standard since release 2.2.0 in May 2009.

- **Jahia**<sup>67</sup> is an integrated web content management and corporate portal server. Jahia Community Edition is now fully integrated with Web 2.0 and Ajax technologies and is leveraging the most used and state of the art J2EE open source libraries (Spring; Lucene; Jackrabbit; Pluto; Hibernate; etc), full multi-language and I18N support; content work-flow; content versioning; Document Management (WebDAV Support); built-in portlet-based interface.
- **GridSphere**<sup>68</sup> portal framework provides an open source portlet based Web portal. Higher-level model for building complex portlets using visual beans and the GridSphere User Interface (UI) tag library.
- **JBoss Portal**<sup>69</sup> includes the portal container and supports a wide range of features including Content Authoring and Publication, Content Search, In Line Editing of Web Content, Editable Content and Site Templates, Content Staging, Internationalization.

#### 4.3.4 Data visualization tools

- **Ajax.org**<sup>70</sup> platform is a pure JavaScript application framework developing real-time collaborative applications that run in the browser.
- **AnyChart**<sup>71</sup> is a flexible Flash based solution that allows to create interactive flash charts. It is a cross-browser and cross-platform charting solution for creating dashboards, reporting, analytics, statistical, financial or any other data visualization schemes.
- **Axiis**<sup>72</sup> is a flexible data visualization framework. Axiis provides both pre-built visualization components as well as abstract layout patterns for data visualization solutions. It's built upon the Degrafa framework and Adobe Flex 3.
- **BirdEye**<sup>73</sup> represents a community project to advance the design and development of a comprehensive open source information visualization and visual analytics library for Adobe Flex. The action script-based library enables users to create multi-dimensional data visualization interfaces for information analysis and presentation.
- **Degrafa**<sup>74</sup> is a declarative graphics framework, licensed under MIT, for creating rich user interfaces, data visualization, mapping, graphics editing etc. Degrafa is currently developed to work in Flex 2 and 3.

<sup>67</sup>[http://www.jahia.org/cms/home/Jahiapedia/How\\_to\\_use\\_Jahia/page247.html](http://www.jahia.org/cms/home/Jahiapedia/How_to_use_Jahia/page247.html)

<sup>68</sup><http://www.gridisphere.org/gridisphere/gridisphere>

<sup>69</sup><http://www.jboss.com/products/platforms/portals/>

<sup>70</sup><http://www.ajax.org>

<sup>71</sup><http://www.anychart.com/home/>

<sup>72</sup><http://www.axiis.org/examples.html>

<sup>73</sup><http://code.google.com/p/birdeye/>

<sup>74</sup><http://www.degrafa.org/samples/data-visualization.html>



- **Chronoscope**<sup>75</sup> can be used to visualize a large number of points of data. Charts can be navigated with the keyboard or mouse.
- **ExtJs**<sup>76</sup> is a cross-browser JavaScript library for building rich web applications. Ext JS supports all major web browsers. It also includes charts.
- **Flex**<sup>77</sup> is an open source framework including chart controls like area, bar, bubble, candlestick, column, HLOCC, Line, Pie, Plot. Flex uses FXG, a graphical interchange format and is used in many other graphical tools to create complex and expressive interactive interfaces.
- **Google Chart Tools**<sup>78</sup> enable adding live charts to a web page. Google Chart Tools consists of two kinds of charts (Image Charts and Interactive Charts) with different features and different requirements to use. These two chart types are based on two different APIs. Google Chart Tools can read live data from a variety of data sources. The Google Chart API returns a chart image in response to a URL GET or POST request. The API can generate many kinds of charts, from pie or line charts to QR codes and formulas. All the information about the charts, such as chart data, size, colors, and labels, are part of the URL.
- **gRaphaël**<sup>79</sup> is a JavaScript library that helps create different types of charts. gRaphaël currently supports Firefox 3.0+, Safari 3.0+, Opera 9.5+ and Internet Explorer 6.0+.
- **JFreeChart**<sup>80</sup> is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications such as bar charts, line charts, pie charts, time series charts, candlestick charts, high/low/open/close charts, wind plots, and meter charts.
- **JQuery Plug-ins**<sup>81</sup> includes many plug-ins for visualization: Visualize by the Filament Group, JQChart, Flot, Sparklines, TufteGraph.
- **Kap IT Labs Diagrammer and Visualizer**<sup>82</sup> offers to Flex community innovative components in 2 areas: data visualization and development optimization. Kap Lab Components are free for non commercial use only.
- **Open Flash Charts**<sup>83</sup> can be used to transform JSON data into different types of Flash Charts.
- **PlotKit**<sup>84</sup> is a chart and graph plotting library for JavaScript. It has support for HTML Canvas and also SVG via Adobe SVG Viewer and native browser support.

<sup>75</sup><http://timepedia.org/chronoscope/>

<sup>76</sup><http://www.sencha.com/products/js/>

<sup>77</sup><http://www.adobe.com/products/flex/>

<sup>78</sup><http://code.google.com/apis/charttools/index.html>

<sup>79</sup><http://g.raphaeljs.com/>

<sup>80</sup><http://www.jfree.org/jfreechart/>

<sup>81</sup><http://jquery.com/>

<sup>82</sup><http://lab.kapit.fr/display/kaplabhome/Home>

<sup>83</sup><http://teethgrinder.co.uk/open-flash-chart-2/>

<sup>84</sup><http://www.liquidx.net/plotkit/>



- **Protovis**<sup>85</sup> composes custom views of data with simple marks such as bars and dots. Protovis defines marks through dynamic properties that encode data, allowing inheritance, scales and layouts to simplify construction.
- **Gephi**<sup>86</sup> is an interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs.
- **XML/SWF Charts**<sup>87</sup> is a simple tool to create attractive charts and graphs from XML data. The XML source can be prepared manually, or generated dynamically using any scripting language (PHP, ASP, CFML, Perl, etc.).
- **dygraphs**<sup>88</sup> is an open source JavaScript library that produces interactive, charts of time series, with zooming functionality. It is designed to display dense data sets and enable users to explore and interpret them.
- **YUI 2: Charts**<sup>89</sup> is used to visualize tabular data on a web page in several possible formats including vertical columns, horizontal bars, lines, and pies. Notable features include support for the DataSource Utility, customizable series and axes, a customizable mouse-over data-tip, combination charts, and skinning.

## 4.4 Relevance feedback

The relevance feedback mechanism uses data mining techniques and performs machine learning on top of the instrumented data for improving different work-flows (plug-in composition schemes). Data mining is an analytic process meant to explore large amounts of data. Its goal is to search consistent patterns and/or relationships between different features, and then to validate the findings by applying the detected patterns to new subsets of data. We will use data mining for prediction, which could be considered the ultimate goal of the data mining process.

The data mining activity can be split into different stages:

1. Data exploration
2. Model construction or pattern identification with validation and verification
3. Deployment i.e. the application of the model to new data in order to generate predictions.

We have depicted in Figure 4.2 a basic flow of the data mining process that includes the steps of data exploration or preprocessing and model construction, applied after a data acquisition and storage process.

### 4.4.1 Data preprocessing

The first step of the data mining process is represented by a thorough analysis of the acquired data. The huge size of acquired data (often several gigabytes or more), the multiple, heterogeneous sources that provide data make today's real-world databases

<sup>85</sup><http://vis.stanford.edu/protovis/>

<sup>86</sup><http://gephi.org/>

<sup>87</sup>[http://www.maani.us/xml\\_charts/](http://www.maani.us/xml_charts/)

<sup>88</sup><http://www.danvk.org/dygraphs/>

<sup>89</sup><http://developer.yahoo.com/yui/charts/>

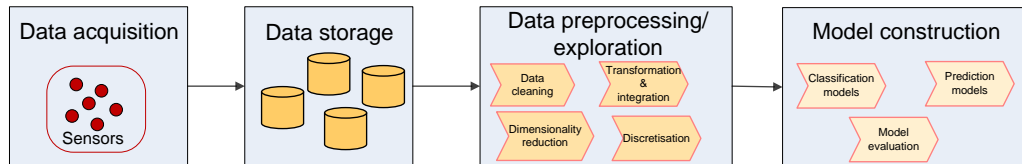


Figure 4.2: Steps in the data mining process

highly susceptible to noisy, missing, and inconsistent data. Low-quality data will lead to low-quality mining results. The data preprocessing stage usually starts with data preparation which may involve data cleaning, transformations, selecting subsets of records and - in case of data sets with large numbers of variables (“fields”) - performing some preliminary feature selection operations to bring the number of variables to a manageable range (depending on the statistical methods which are being considered).

**Data cleaning** is a process that deals with filling in the missing values, smoothing out noise, identifying outliers, and correcting inconsistencies in the data.

The **missing values** problem can be approached in several ways [6]:

- Ignore the tuple – not an effective method, unless the tuple contains several attributes with missing values.
- Fill in the missing value manually – a time-consuming method not feasible given a large data set with many missing values.
- Use a global constant to fill in the missing value – this method replaces all missing attribute values by the same constant, such as a label like “Unknown” or infinite.
- Use the attribute mean to fill in the missing value.
- Use the attribute mean for all samples belonging to the same class as the given tuple.
- Use the most probable value to fill in the missing value. This may be determined with regression, inference-based tools using a Bayesian formalism, or decision tree induction.

**Noise** is a random error or variance in the measured values of the attributes. Given numerical attributes there are some smoothing techniques that can be applied to reduce the noise [6]:

- Binning – smooth a sorted data value by analyzing its neighborhood that is, the values around it. The sorted values are distributed into a number of buckets or bins. In smoothing by bin means each value in a bin is replaced by the mean value of the bin. Similarly, in smoothing by bin medians each bin value is replaced by the bin median. In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries.
- Regression – the attributes values can be smoothed by fitting the data to a function. Linear regression involves finding the “best” line to fit two attributes, so that one attribute can be used to predict the other. Multiple linear regression is an extension of linear regression, where more than two attributes are involved and the data are fit to a multidimensional surface.



- Clustering – outliers may be detected by clustering, a process in which similar values are organized into groups (clusters). The values that fall outside of the set of clusters may be considered outliers.

Missing values, noise, and inconsistencies contribute to inaccurate data [6]. The first step in data cleaning as a process is discrepancy detection. Discrepancies may arise from inconsistent data representations and the inconsistent use of codes. Errors in instrumentation modules that acquire data, and system errors, are another source of discrepancies. Errors can also occur when the data are inadequately used for purposes other than originally intended. There are a number of different tools that can aid in the discrepancy detection. Some data inconsistencies may be corrected manually using external references. Most errors, however, will require **data transformations**. This is the second step in data cleaning as a process. That is, once the discrepancies are found, a series of transformations are typically defined and applied for correcting them.

Data mining often requires **data integration** – the merging of data from multiple data stores. The data may also need to be transformed into appropriate forms for mining. Data integration combines data from multiple sources into a coherent data store, as in data warehousing. These sources may include multiple databases, data cubes, or flat files. There are some issues to consider during data integration. How can equivalent real-world entities from multiple data sources be matched up? This is referred to as the entity identification problem. Redundancy is another important issue. An attribute may be redundant if it can be derived from another attribute or set of attributes. Some redundancies can be detected by correlation analysis. Given two attributes, such analysis can measure how strongly one attribute implies the other, based on the available data. For numerical attributes, the correlation between two attributes can be obtained by computing the correlation coefficient. A third important issue in data integration is the detection and resolution of data value conflicts. This may be due to differences in representation, scaling, or encoding. When matching attributes from one database to another during integration, special attention must be paid to the structure of the data. This is to ensure that any attribute functional dependencies and referential constraints in the source system match those in the target system.

Data is then transformed or consolidated into appropriate forms for mining. Data transformation involve the following [6]:

- Smoothing – removes noise from the data. Such techniques include binning, regression, and clustering.
- Aggregation – applies summary operations on the data. This is typically used in constructing a data cube for analysis of the data at multiple granularities.
- Generalization – replaces low-level or primitive (raw) data by higher-level concepts through the use of concept hierarchies.
- Normalization – scales the attribute data so as to fall within a small specified range.
- Attribute construction (or feature construction) – constructs and adds new attributes from the given set of attributes to help the mining process. By combining attributes missing information about the relationships between data attributes can be discovered.



Complex data analysis and mining on huge amounts of data can take a long time, making such analysis is impractical or infeasible. **Data reduction** techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. The strategies for data reduction are [6]:

- Data cube aggregation – data cubes created for varying levels of abstraction are often referred to as cuboids, so that a data cube may instead refer to a lattice of cuboids. Each higher level of abstraction further reduces the resulting data size.
- Attribute subset selection – detects and removes irrelevant, weakly relevant, or redundant attributes or dimensions. The goal of attribute subset selection is to find a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes.
- Dimensionality reduction – data encoding or transformations are applied in order to obtain a reduced or compressed representation of the original data. There are two popular and effective methods of lossy dimensionality reduction: wavelet transforms and principal components analysis.
- Discretization and concept hierarchy generation – replaces raw data values for attributes by ranges or higher conceptual levels. The most used methods that can be applied for discretization are: (1) binning: a top-down splitting technique based on a specified number of bins; (2) histogram analysis: partitions the values for an attribute into disjoint ranges; (3) entropy-based discretization: explores class distribution information in its calculation and determination of split-points (data values for partitioning an attribute range) using energy functions; (4) cluster analysis: partitions the objects into groups based on a similarity function; (5) discretization by intuitive partitioning: achieves numerical ranges partitioned into relatively uniform, easy to read intervals that appear intuitive or natural.

#### 4.4.2 Model construction and deployment

The **model construction** consists in the performance analysis of several algorithms and the choice of the best models suitable for the interest problem. This analysis is encountered in the literature under two different forms: classification and prediction. **Model deployment** involves the application of the best classification or prediction model to new data in order to generate predictions or estimates of the expected outcome.

#### Classification models

Classification is the operation of separating various entities into several classes [7]. It requires a deep understanding of the nature of data that is provided and a concise determination of the classification problem suited for the given data: determine and define how many target classes there are, define the boundaries of each class in terms of the input variables, construct a set of decision rules from class boundaries to define each class, determine the prior probability of each class, and if appropriate determine the cost of making the wrong choice in assigning samples to a given class.

As presented by [7], [6], [8] the most explored and used supervised classification algorithms are:



- *Nearest-Neighbor Classifiers*: find in the  $N$ -dimensional feature space the closest object from the training set to an unknown object being classified.
- *Decision Tree Induction*: represents the learning of decision trees from class-labeled training tuples. A decision tree is a flowchart-like tree structure, where each internal node (non-leaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label.
- *Bayesian Networks*: take the form of statistical classifiers. They can predict class membership probabilities using Bayes rules.
- *Neural Networks*: represent a set of connected input/output units in which each connection has a weight associated with it. During the learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of the input tuples. They require a number of parameters that are typically best determined empirically, such as the network topology or ‘structure’.
- *Random Forests and Boosting*: the random forest algorithm trains a number of trees on slightly different subsets of data (bootstrap sample), in which a case is added to each subset containing random selections from the range of each variable. This group of trees are similar to an ensemble. Each decision tree in the ensemble votes for the classification of each input case. Boosting is an ensemble method that combines several decision rules (weak classifiers) based on an error minimization criteria.
- *Support Vector Machines (SVM)*: are a set of related methods for supervised learning, applicable to both classification and regression problems. A SVM classifier creates a maximum-margin hyperplane that lies in a transformed input space and splits the example classes, while maximizing the distance to the nearest cleanly split examples. The parameters of the solution hyperplane are derived from a quadratic programming optimization problem.
- *Other supervised classification algorithms* include genetic algorithms, rough set methods, fuzzy set approaches.

## Prediction models

Whereas classification predicts categorical (discrete, unordered) labels, numeric prediction represents the task of predicting continuous (or ordered) values for given input [6]. For example, we can build a classification model to categorize images recorded by a stereo system mounted on a vehicle as either containing pedestrians or not, or a prediction model to find the CPU load of a parallel database system given the input query.

The most widely used approach for numeric prediction is regression. It is used to model the relationship between one or more independent or predictor variables and a dependent or response variable (which is continuous-valued). Many problems can be solved by linear regression, and even more can be tackled by applying transformations to the variables so that a nonlinear problem can be converted to a linear one. As described by [7], [6] regression analysis can take the forms of: linear regression and nonlinear regression. Several classification algorithms such as: neural networks, boosting, SVM can be configured to perform regression, hence they can be used for prediction.



## Evaluation models

We have presented a variety of classification and prediction models. The question that comes next is how do we choose the best model for our particular problem. The answer may sound simple but in fact, it sometimes involves a very elaborate process. There are many techniques to evaluate the performance of the classification and prediction models.

The process of modeling and evaluation is iterative. Often, the evaluation process can point out some improvement issues that may lead to modifications in the data preprocessing and modeling stage. The modifications may help to enhance the predictability of the model. Thus a complete sequence of iterative operations may be: modeling, evaluation, improvement.

Among the most popular metrics for evaluating classification and prediction we can mention [6]: error rate or miss-classification rate, confusion matrix, sensitivity and specificity, precision and recall, loss function, generalization error, relative error, ROC curve, learning curve. An estimate of the classifier's accuracy can be obtained by: random sub-sampling, cross validation, bootstrapping.

## Model deployment

The last step in the data mining process involves the application of the best classification or prediction model to new data in order to generate predictions or estimates of the expected outcome. After a satisfactory model or set of models has been trained for a particular application we want to apply them in order to obtain quickly classification information for new data.

### 4.4.3 Data mining for predicting system performance

The major role of the relevance feedback mechanism is to predict values for some of the metrics proposed in [2]. Most of the metrics refer to system performance (CPU Load, data transfer, execution time). Hence a detailed analysis of state of the art work related to data mining approaches for system performance prediction is required.

Most of the work concerns predicting the application execution time and performance based on different features and using various prediction algorithms. Multilayer neural networks are used by [9] for predicting performance of application run-times for two parallel applications: SMG 2000, a semi-coarsening multi-grid solver, and HPL, an open source implementation of LINPACK.

Statistical machine learning techniques (Kernel Canonical Correlation Analysis) are used by [10] for modeling system performance. Their methodology extracts correlations between a workload's pre-execution characteristics or configuration parameters and post-execution performance observations. The validation is done on three different storage and compute based parallel systems.

A framework for extraction and prediction of on-line workload data from a workload manager of a mainframe operating system (z/OS) is proposed by [11]. They use recursive feature elimination as feature selection method and self learning models such as neural network or support vector regression.

Other approaches regard predicting system crashes in parallel, distributed or serial applications or in the context of the operating system. System level metrics and linear regression and decision trees are used in [12] to predict web server crashes. Support Vector Machine algorithms are utilized by [13] to predict computer system failure events based on log files.



#### 4.4.4 Data mining tools

In what follows we will present some tools that are widely used in the data mining process nowadays.

##### **The Weka machine learning**

The Weka workbench [14] is a collection of state-of-the-art machine learning algorithms and data preprocessing tools.

###### *Functionality*

The workbench includes methods for all the standard data mining problems: regression, classification, clustering, association rule mining, and attribute selection. It is also well suited for developing new machine learning schemes. It is designed so that you can quickly try out existing methods on new datasets in flexible ways. It provides extensive support for the whole process of experimental data mining, including preparing the input data, evaluating learning schemes statistically, and visualizing the input data and the result of learning.

###### *Usability*

One way to use Weka is through a graphical user interface called the “Explorer”. This gives access to all of its facilities using menu selection and form filling. For example, it quickly reads a dataset from an ARFF file (or spreadsheet) and builds a decision tree from it. There are two other graphical user interfaces to Weka. The “Knowledge Flow” interface allows to design configurations for streamed data processing and the “Experimenter” interface is designed to help answering a basic practical question when applying classification and regression techniques: which methods and parameter values work best for the given problem? The algorithms can either be applied directly to a dataset through the GUI interface or called from the Java code.

###### *License*

The Weka system is an open source software written in Java and distributed under the terms of the GNU General Public License. It runs on almost any platform and has been tested under Linux, Windows, and Macintosh operating systems.

###### *Advantages and Disadvantages*

First, it is open source, which not only means that it can be obtained free, but more importantly it is maintainable, and modifiable, without depending on the commitment, health, or longevity of any particular institution or company. Second, it provides a wealth of state-of-the-art machine learning algorithms that can be deployed on any given problem. It is fully implemented in Java and runs on almost any platform.

A fundamental disadvantage of the Weka Explorer is that it holds everything in main memory when you open a dataset, it immediately loads it all in. This means that it can only be applied to small to medium-sized problems. If larger datasets are to be processed, some form of sub-sampling is generally required. A second disadvantage is the flip side of portability: a Java implementation is generally slower than an equivalent in C/C++.

##### **LibSVM**

LibSVM [15] is a library for support vector machines (SVM). Its goal is to allow users to easily use SVM as a tool.

###### *Functionality*

LibSVM is an integrated software for support vector classification, (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). It



supports multi-class classification. LibSVM provides a simple interface where users can easily link it with their own programs. Main features of LIBSVM include: different SVM formulations, efficient multi-class classification, cross validation for model selection, probability estimates, weighted SVM for unbalanced data.

#### *Usability*

The LibSVM has both C++ and Java sources, a user interface demonstrating SVM classification and regression. It has Python, R (also Splus), MATLAB, Perl, Ruby, Weka, Common LISP, CLISP, Haskell and LabVIEW interfaces. It's also included in some data mining environments such as RapidMiner and PCP.

#### *License*

The LibSVM license (“the modified BSD license”) is compatible with many free software licenses such as GPL. Hence, it is very easy to use LibSVM in software applications. It can also be used in commercial products by clearly indicate that LibSVM is used and retain the LibSVM copyright file in the software.

#### *Advantages and Disadvantages*

SVMs gain flexibility by introducing the kernel. No assumptions about the functional form of the transformation, which makes data linearly separable, is necessary. SVMs provide a good out-of-sample generalization and can be robust, even when the training sample has some bias. SVMs deliver a unique solution, since the optimality problem is convex.

A common disadvantage of non-parametric techniques such as SVMs is the lack of transparency of results.

## **RapidMiner**

Rapidminer (Rapid-I GmbH, Dortmund, Germany) is an open source environment for data mining. The software is written in Java. The functionality of RapidMiner can be accessed through a friendly GUI or programatically by using Java. There are numerous build in algorithms from statistical learning field along with machine learning, unsupervised learning, planning, etc. RapidMiner has a good connectivity with various data sources including database access.

RapidMiner can be extended with various modules. There are modules that connect RapidMiner with R and with Weka environment. Being written in Java the RapidMiner elements can be integrated in any Java written software through provided API.

Because the company that produces RapidMiner sells a commercial license of the RapidMiner, this software is advertised more in the business/industry environment rather than in research communities. This aspect is visible into the functionality of the software which is more oriented in eye catching interface rather than on functionality. RapidMiner is intended to be an interactive software, where user can dynamically change an experiment and visualize (or preview) the results immediately. This might not be the best approach in the research community where one wants flexible automated tools which will process large amount of data without human intervention. Parallelism can be achieved at data level.

## **Matlab**

Matlab (MathWorks, Massachusetts USA) is proprietary software that is optimized for algebra calculus. One can find a wide range of tools, starting from basic algebra to differential equation solvers, statistics, etc. In Matlab one can integrate various



modules (financial, image processing, interpolation, etc) in order to extend its functionality. The extension module is called a toolbox. There are no dedicated tools for data mining, but one can find implementations for major algorithms found in data mining field. The usual statistical functions are integrated in the medium. One can find toolboxes for neuronal networks, curve fitting, model based calibration, etc. Matlab can be used as an integration environment. The mathematical language is pretty straight forward but the general purpose control structures are implemented inefficiently. The language is an interpreted one, but there are available some tools that generate standalone packages that can be executed on other machines. In the same time the integration with other tools is difficult. There are some means of calling Matlab functions from C but the mechanism is rigid and has a long learning curve. One can use command line in order to call Matlab scripts, or execute various commands from Matlab but there is no easy way to communicate with external programs (shared memory, pipes, function calls, etc). MathWorks provide a separate toolbox for parallel computing. The parallelization can be achieved at data level relatively easy without any additional software (or costs) but its performance might not be as good as when using the dedicated toolbox. A major advantage of Matlab is that it has a large variety of visualization tools for almost any kind of data.

### Other tools

Numerous tools that perform classification, prediction or data preprocessing are available. We have included a short summary of these tools in Table 4.1.

Tool	Web Page	Feature Selection	Clustering	Classification	Regression
R-Project	<a href="http://cran.r-project.org">cran.r-project.org</a>		•	•	•
MultiBoost	<a href="http://multiboost.org">multiboost.org</a>			•	
Dlib	<a href="http://dlib.net">dlib.net</a>		•	•	•
redsvd	<a href="http://code.google.com/p/redsvd">code.google.com/p/redsvd</a>	•		•	
jblas	<a href="http://jblas.org">jblas.org</a>	•			
Orange	<a href="http://ailab.si/orange">ailab.si/orange</a>	•		•	
Mulan	<a href="http://mulan.sourceforge.net">mulan.sourceforge.net</a>			•	•
Waffles	<a href="http://waffles.sourceforge.net">waffles.sourceforge.net</a>		•	•	•
ELF	<a href="http://elf-project.sourceforge.net">elf-project.sourceforge.net</a>			•	•
MDP	<a href="http://mdp-toolkit.sourceforge.net">mdp-toolkit.sourceforge.net</a>	•	•	•	
JProGraM	<a href="http://dii.unisi.it/~freno">dii.unisi.it/~freno</a>		•	•	
Shogun	<a href="http://shogun-toolbox.org">shogun-toolbox.org</a>	•		•	•
LibG	<a href="http://libagf.sourceforge.net">libagf.sourceforge.net</a>		•	•	
OpenKernel	<a href="http://openkernel.org">openkernel.org</a>			•	

Table 4.1: Other Tools Functionalities

The ‘•’ denotes the fact that the respective feature is supported by the corresponding tool.



## 5. Conclusion

This deliverable reported on the work that has been done in LarKC in order to identify what existing tools and technologies can be potentially used for building an instrumentation and monitoring solution for LarKC plug-ins, work-flows and platform. Furthermore, in this deliverable we gathered requirements for instrumentation and monitoring of large scale reasoning platforms such as LarKC.

Based on the analysis of instrumentation tools and the identification of requirements for code instrumentation we were able to select AspectJ as a very promising solution for instrumentation that we will further explore and use in the LarKC instrumentation. AspectJ provides the best foundation for building an instrumentation solution that fulfills the requirements identified in Section 3.2.1. Regarding monitoring tools and technologies, the approaches surveyed in Section 4.2 cover to a certain percent the needs for monitoring identified in Section 3.2.2 and Section 3.2.3. However there is no solution that can be simply taken off the shelf and used for LarKC monitoring of plug-ins, work-flows and platform. From a conceptual point of view, approaches such as OpenNMS or Zabbix provide good inspirational models. Most of the tools however have a different focus being either centered on monitoring of very low metrics or being developed for specific domains (e.g. for web applications). Furthermore none of them handles and stores data in a semantic format (i.e. RDF triples) as it will have to be the case for LarKC monitoring. Another sensitive issue is the license issue, most monitoring tools that we analyzed being distributed under a different license than Apache License, the license used in LarKC. As expected, there exists a multitude of tools for visualization, analysis and reporting on monitored data. Such tools have been developed using various technologies and frameworks, such as web development frameworks, content management systems and portals. The visualization, analysis and reporting component that will be build for LarKC monitoring data will have to fulfill the requirements identified in Section 3.2.4, in particular to provide an intuitive, easy to use, real-time solution and handle a very large amount of data. Few of the visualization tools that we investigated fulfill many of these requirements, and none of them all. Finally, our analysis of existing methods and approaches for relevances feedback based on identified requirements for this component in LarKC settings, showed that there exist many potential tools for building a relevance feedback component that uses data mining techniques and performs machine learning on top of the instrumented data for improving different work-flows. Approaches such as Weka and libSVM are some of the approaches that we plan to experiment for the relevance feedback component.

The initial findings and insights from this deliverable will be considered as input for T11.2, design and implementation of the first version of instrumentation and monitoring platform. Deliverable D11.2, due M35 will report on how the requirements identified in the current deliverable were taken into account and how the tools and methods surveyed in this deliverable, that were identified as relevant, are being used in the first version of the instrumentation and monitoring platform.



## REFERENCES

- [1] LarKC, “D1.2.1 initial operational framework,” Tech. Rep., 2008.
- [2] I. Toma, R. Brehar, S. Bota, M. Negru, A. Vatavu, and M. Chezan, “D11.1.2: Larkc metrics ontologies,” LarKC Project Deliverable, Tech. Rep. D11.1.2, 2010.
- [3] E. Zudilova-Seinstra, T. Adriaansen, and R. v. Liere, “Overview of interactive visualisation,” in *Trends in Interactive Visualization*, ser. Advanced Information and Knowledge Processing, L. Jain, X. Wu, R. Liere, T. Adriaansen, and E. Zudilova-Seinstra, Eds. Springer London, 2009, pp. 1–13, 10.1007/978-1-84800-269-1. [Online]. Available: <http://dx.doi.org/10.1007/978-1-84800-269-2>
- [4] M. Friendly, “Milestones in the history of thematic cartography, statistical graphics, and data visualization,” 2009, <http://datavis.ca/milestones/>. [Online]. Available: <http://www.math.yorku.ca/SCS/Gallery/milestone/>
- [5] S. G. Eick, “Graphically displaying text,” *Journal of Computational and Graphical Statistics*, vol. 3, pp. 127–142, 1994.
- [6] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2006.
- [7] R. Nisbet, J. Elder, and G. Miner, *Handbook of Statistical Analysis and Data Mining Applications*. Academic Press, 2009.
- [8] O. Maimon and L. Rokach, *Data Mining and Knowledge Discovery Handbook*. Springer, 2010.
- [9] K. Singh, E. İpek, S. A. McKee, B. R. de Supinski, M. Schulz, and R. Caruana, “Predicting parallel application performance via machine learning approaches: Research articles,” *Concurr. Comput. : Pract. Exper.*, vol. 19, no. 17, pp. 2219–2235, 2007.
- [10] A. S. Ganapathi, “Predicting and optimizing system utilization and performance via statistical machine learning,” Ph.D. dissertation, EECS Department, University of California, Berkeley, Dec 2009. [Online]. Available: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-181.html>
- [11] M. Bensch, D. Brugger, W. Rosenstiel, M. Bogdan, W. G. Spruth, and P. Baeuerle, “Self-learning prediction system for optimisation of workload management in a mainframe operating system,” in *ICEIS (2)*, 2007, pp. 212–218.
- [12] J. Alonso, J. Torres, and R. Gavalda, “Predicting web server crashes: A case study in comparing prediction algorithms,” in *ICAS '09: Proceedings of the 2009 Fifth International Conference on Autonomic and Autonomous Systems*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 264–269.
- [13] E. W. Fulp, G. A. Fink, and J. N. Haack, “Predicting computer system failures using support vector machines,” in *WASL'08: Proceedings of the First USENIX conference on Analysis of system logs*. Berkeley, CA, USA: USENIX Association, 2008, pp. 5–5.



- 
- [14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The weka data mining software: An update,” *SIGKDD Explorations*, vol. 11, 2009.
- [15] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.