



## **LarKC**

*The Large Knowledge Collider*

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

**FP7 – 215535**

---

# **D6.3 Urban Computing environment specification**

---

**Coordinator: Emanuele Della Valle**

**With contributions from: Daniele Dell'Aglio, Irene Celino,  
Kono Kim**

**Quality Assessor: Georgina Gallizo**

**Quality Controller: Emanuele Della Valle**

Document Identifier:	LarKC/2008/D6.3/V1.0
Class Deliverable:	LarKC EU-IST-2008-215535
Version:	version 1.0
Date:	May 28, 2009
State:	final
Distribution:	public



## EXECUTIVE SUMMARY

In the description of work of the LarKC project, this deliverable was indicated as “Urban Computing environment specification” and was aimed at describing the *design* of a Urban Computing application making use of the LarKC platform to perform its functionalities. However, the work on the implementation of the LarKC platform (in WP5) and also some activities related to the development of some plug-ins (in WP2-WP3-WP4) proceeded in the direction of a first public release of the whole platform within M14. For those reasons, in the Bled meeting WP6 proposed to the TMB to turn this deliverable into the *realization* of a “mock-up” of the Urban Computing application we had in mind, that we called “*Urban Baby LarKC*”. The TMB accepted the proposal and a joint WP5-WP6 task force was set up.

Moreover, with the help of the platform and plug-ins developers, we not only developed a “mock-up” but we were able to achieve much more: some specific plug-ins were designed and rapidly prototyped and some simple pipelines were instantiated and executed over the available code of the LarKC platform. In this way, we not only took an important step towards the actual realization of a fully-fledged Urban Computing application, but, playing the role of “early adopters” of the technologies developed within LarKC, we also had the chance to face with all the problems of an initial experimentation with the platform and, as a consequence, to provide early feedback, lessons learned and new requirements to the platform and plug-in developers.

This document reports on the activities performed towards the Urban Baby LarKC: the data preparation; the development of some sample applications using those data; the realization of some applications to be used as “term of comparison” for the Urban Baby LarKC; the plug-ins we developed for our scenario; the pipelines we constructed with those plug-ins; and the client application to launch the functionalities from a map GUI. In the final part of the deliverable, we also provide a summary of the findings of this early adoption activity: lessons learned, feature requests, new requirements for the platform and the plug-in design and development.

Apart from reporting on what we already achieved within WP6, this document aims at representing the *initial step of the road-map* towards a fully fledged Urban Computing application developed with LarKC technologies and tools. In fact, in order to realize a complete and meaningful scenario, the current LarKC platform is still in progress, and therefore in a non stable status and subject to changes; however, following an incremental approach and the so-called “evolutionary prototyping” development, with the Urban Baby LarKC we wanted to demonstrate the feasibility of our plans. By adding more functionalities and by complicating the scenario (and its respective pipeline), we will be able also to demonstrate the advantages of the LarKC approach with regards to existing technologies.

In the future deliverables of WP6 – namely D6.5 “Urban Computing environment v1”, D6.8 “Urban Computing environment v2” and D6.10 “Urban Computing environment v3” – we will present the advancements of our work towards the realization of a series of Urban Computing demonstrators of increasing complexity that take advantage of the more and more mature implementation of the LarKC platform and the richer support it will offer for the plug-ins development and the pipelines construction and execution.



## 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>	6.3	<b>Title</b>	Urban Computing environment specification
<b>Work Package</b>	<b>Number</b>	6	<b>Title</b>	Urban Computing

<b>Date of Delivery</b>	<b>Contractual</b>	M14	<b>Actual</b>	31-May-09
<b>Status</b>	version 1.0		final	<input checked="" type="checkbox"/>
<b>Nature</b>	prototype <input checked="" type="checkbox"/> report <input type="checkbox"/> dissemination <input type="checkbox"/>			
<b>Dissemination Level</b>	public <input checked="" type="checkbox"/> consortium <input type="checkbox"/>			
















<b>Authors (Partner)</b>	Daniele Dell’Aglío (Cefriel), Irene Celino (Cefriel), Kono Kim (Saltlux)			
<b>Resp. Author</b>	Emanuele Della Valle		<b>E-mail</b>	emanuele.dellavalle@cefriel.it
	<b>Partner</b>	Cefriel	<b>Phone</b>	+39 (02) 23954-324

<b>Abstract (for dissemination)</b>	Report on the first release of the “Urban Baby LarKC”, i.e. the realization of a Urban Computing scenario over the LarKC platform, and the lessons we learned from this experience of early adoption of LarKC technology.
<b>Keywords</b>	Urban Computing, Urban Baby LarKC, lessons learned

Version Log			
Issue Date	Rev No.	Author	Change
April 9, 2009	1	Irene	Document initialization
April 17, 2009	2	Irene	ToC structure
April 24, 2009	3	Irene	Report on activities performed
May 7, 2009	4	Irene	Lessons learned
May 11, 2009	5	Daniele	Data preparation explanation
May 13, 2009	6	Daniele	Description of plug-ins and pipelines
May 14, 2009	7	Irene	Executive summary and Introduction
May 15, 2009	8	Daniele	Pipelines recap and client
May 15, 2009	9	Irene	Conclusion
May 26, 2009	10	Emanuele	Some rephrasing and more precise information
May 28, 2009	11	Irene	Document review on the basis of the QA feedbacks



## 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 EKSPERIMEN- TALNI 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



## TABLE OF CONTENTS

LIST OF FIGURES	6
1 INTRODUCTION	7
2 URBAN COMPUTING IN LARKC: THE STORY SO FAR	8
2.1 Data Preparation . . . . .	8
2.1.1 AMA data sets . . . . .	9
2.1.2 Data sets from Web . . . . .	9
2.2 Browser for Geo-data . . . . .	10
2.3 Term of comparison . . . . .	11
2.4 Urban Baby LarkC plug-ins . . . . .	11
2.5 Urban Baby LarkC pipelines . . . . .	12
2.5.1 First pipeline . . . . .	13
2.5.2 Second pipeline . . . . .	13
2.5.3 Third pipeline . . . . .	15
2.5.4 Recap . . . . .	16
2.6 Urban Baby LarkC client . . . . .	19
3 LESSONS LEARNED FROM THE EARLY ADOPTION OF LARKC	20
4 CONCLUSIONS AND NEXT STEPS	24
REFERENCES	25



## LIST OF FIGURES

2.1	Screenshot of the RDF browser for geographical data. . . . .	10
2.2	First Urban Baby LarKC pipeline . . . . .	13
2.3	Second Urban Baby LarKC pipeline . . . . .	14
2.4	The complex selection strategy . . . . .	14
2.5	Third Urban Baby LarKC pipeline . . . . .	15
2.6	Urban Baby LarKC Client . . . . .	19



## 1. Introduction

This deliverable does not simply represent the “Urban Computing environment specification”, i.e. the *design* of a Urban Computing application, but it consists in the high-level report on the activities towards the so-called “*Urban Baby LarKC*”, i.e. the *implementation* of an application that makes use of the LarKC platform and of some plug-ins to concretize some Urban Computing scenario.

In this way, we not only took an important step towards the actual realization of a fully-fledged Urban Computing application, but, playing the role of “early adopters” of the technologies developed within LarKC, we also had the chance to face with all the problems of an initial experimentation with the platform and, as a consequence, to provide early feedback, lessons learned and new requirements to the platform and plug-in developers.

Due to the immaturity of the platform, the scenario we are currently able to demonstrate in our Urban Baby LarKC is limited; nonetheless, it is the first step of the more complex scenario we envisioned in deliverable D6.1 [2] and in [8, 7]. This scenario is about a user that wants to plan his movement, by possibly using a combination of transportation means, towards a particular destination that can be a known place (simple scenario) or some dynamically-chosen goal place (more complex scenario), like a monument – selected by the relevant ones of a city –, an event – among those published on the Web – or a friend – whose position can change over time. In order to fulfill the user request, numerous distributed and heterogeneous data sources should be accessed and several different parameters should be taken into account to calculate the “most desirable” path for the user; therefore, this is a good scenario to demonstrate the advantages of using LarKC technologies over traditional solutions and custom implementations. At the time of writing (May 2009), we were able to implement with LarKC only the simple scenario, but we are confident that we will be shortly able to add more functionalities to realize also the more complex scenario.

The document is structured as follows. Chapter 2 reports on the activities performed towards the Urban Baby LarKC: in particular, Section 2.1 is about the data preparation; Section 2.2 describes the development of some sample applications using those data; Section 2.3 introduces the realization of some applications to be used as “term of comparison” for the Urban Baby LarKC; Section 2.4 and Section 2.5 give some insight respectively on the plug-ins we developed for our scenario and on the pipelines we constructed with those plug-ins; finally, Section 2.6 illustrates the client application to launch the functionalities from a map GUI. In Chapter 3, we provide a summary of the findings of this early adoption activity: lessons learned, feature requests, new requirements for the platform and the plug-in design and development; the deliverable is concluded by Chapter 4 which draws some conclusions and foresees the next steps.

Those future works will be the topic of the future deliverables of WP6 – namely D6.5, D6.8 and D6.10 – in which we will present the advancements of our work towards the realization of a series of Urban Computing demonstrators of increasing complexity that take advantage of the more and more mature implementation of the LarKC platform and the richer support it will offer for the plug-ins development and the pipelines construction and execution.



## 2. Urban Computing in LarKC: the story so far

In order to get to a concrete implementation of a Urban Computing scenario over the LarKC platform, several steps must be taken, from the data preparation, through the design of the system, till the realization of the prototype.

In this chapter, we provide a short summary of the activities performed with WP6 to get to the first public release of the Urban Baby LarKC, the first prototype of WP6. We do not provide small details on tasks and implementation, since our purpose in this document is to give an overview of the general process needed to develop a Urban Computing application with LarKC. More details are available, however, on the project wiki and in the code documentation and some sample applications are available online, as explained in the following; of course, further explanations can be provided upon request.

In the path towards the Urban Baby LarKC we also got some early lessons learned and feature requests for LarKC, which are provided in the following chapter.

### 2.1 Data Preparation

Since the beginning of the project, we are collecting pointers to data sources of interest to be used in Urban Computing scenarios; in deliverable D6.1 [2], we provided a first list of interesting data sources.

We started analyzing the available data sets, in order to select the first ones to be consider in the Urban Baby LarKC and in order to understand their peculiarities and their characteristics. We decided to go into more depth of a subset of the data sources that we have found in the initial survey and in deliverable D6.4 [3] we provide information about these sources. We can classify the datasets we are considering in two classes:

- **AMA DATA SETS:** this data, in GIS-specific format, contains information about the city of Milano as like topology and traffic information released by *Agenzia Milanese Mobilità Ambiente*<sup>1</sup> (in english: “Milan Mobility and Environment Agency”);
- **DATA SETS FROM WEB:** the data sets available on the Web. Some of these sources belong to the Linking Open Data (LOD) cloud and they contain links to other data sets. Other sources do not have explicit links but we think that will be possible to create connections to other ones.

In the selection of the data sources we take into account different characteristics: data sets can contain static or dynamic data, the format of the data could be RDF or convertible ones (for example XML documents, that can be translated in RDF with GRDDL), data sets can have links to other sources or not to have them (but they can be generated) and so on. In this way we have an heterogeneous set of data sets that should be treated in different ways. In the future LarKC should be able to treat some different kind of data formats and their conversion in RDF; currently we manually prepared the data in order to use this data within the current LarKC platform. In the following we will briefly describe several steps that were taken to treat the two sets of data sources.

---

<sup>1</sup><http://www.ama-mi.it>



### 2.1.1 AMA data sets

As reported in D6.4 [3] the data obtained from AMA is not in an RDF format; in order to use this data with LarKC we converted it in RDF and we exposed it in different ways. More precisely, the main steps of the AMA data preparation are:

- **ONTOLOGY MODELING:** on the basis of the analysis, we derived some ontological schemata to represent the data; details and FAQs about the ontologies we modeled are available on the project wiki at <http://wiki.larkc.eu/LarkcProject/WP6/WorkInProgress/AMAData>;
- **SPARQL ENDPOINT:** in order to access the selected data sources as SPARQL endpoint, we used a mapping tool – namely D2R [1] – and configured it to expose the data on the Web and have the possibility to query them; the D2R server is available on line at <http://seip.cefriel.it/ama/2>;
- **RDF DUMPS:** the SPARQL endpoint realized with D2R do not convert the original data, but just wraps the data source; however, D2R offers some facilities to get RDF dumps out of the wrapped data source; we exploit those facilities and we obtained several RDF dumps: one which included the whole graph of Milano roads and a bunch of others which contain meaningful subsets of the whole graph;
- **ALLEGROGRAPH ENDPOINT:** since the Urban Computing scenario involves geographical data, we looked for tools that are able to deal and to easily query RDF data that include location information; AllegroGraph<sup>3</sup> is an RDF store which offers some facilities to deal with geographic information; we used AllegroGraph to have a SPARQL endpoint to query the RDF data and to get the possibility to geo-spatially index the data; therefore, the data can be also queried with a custom SPARQL extension which allows for selecting RDF triples on the basis of the location information<sup>4</sup>.

### 2.1.2 Data sets from Web

Among the available data sets on the Web that we considered there are some sources that expose RDF data: in that cases we examined related documentation, schemata and we look into the available methods to access the data.

In the other cases the data sources that we considered allow to retrieve little amount of data through Web services with an XML output. It means that we cannot get all the data contained in the sources (and convert them as we did for AMA data sets), but we have to perform the transformation on the fly. The main steps of this process are:

- **ONTOLOGY MODELING:** every service uses a different custom schema to format its data; first of all we examined the kind of information and the structure of

---

<sup>2</sup>The access to the D2R server is password-protected. Interest readers can get proper credentials at <http://wiki.larkc.eu/LarkcProject/WP6/Private/AmaSparqlPwd> or ask for them via email to [daniele.dellaglio@cefriel.it](mailto:daniele.dellaglio@cefriel.it).

<sup>3</sup>AllegroGraph RDFStore Web 3.0's Database <http://agraph.franz.com/allegrograph/>

<sup>4</sup>AllegroGraph SPARQL geo-extension <http://franz.com/agraph/support/documentation/current/sparql-geo.html>

the output. Then we looked for available vocabularies with suitable terms to represent in RDF the data we receive from the Web service;

- **CONVERSION PREPARATION:** in order to perform the conversion from XML to RDF it's possible to follow the GRDDL approach, that uses algorithms defined in XSLT format.

## 2.2 Browser for Geo-data

Once we obtained a meaningful set of data sources and realized the possibility to query them in SPARQL, we started testing and employing the data to implement demonstrators towards our idea of Urban Computing with Semantic Technologies.

The first tangible result was a visual RDF browser for geographical data. It is an application built on top of Google Maps that lets the user explore the data sources by “navigating” across the data in a natural and intuitive way, i.e. by moving on a map. This browser is available on line<sup>5</sup> at <http://seip.cefriel.it/amabrowser/> and a screenshot of its functioning is offered in Figure 2.1.

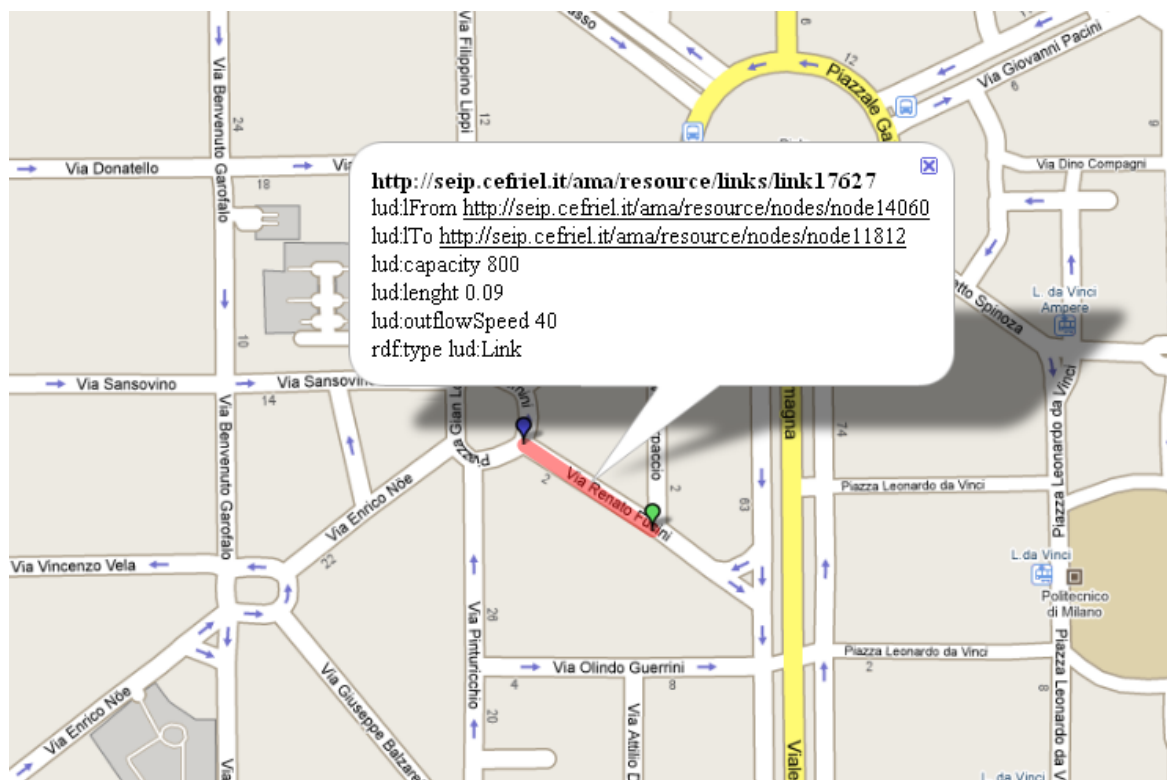


Figure 2.1: Screenshot of the RDF browser for geographical data.

<sup>5</sup>The access to the RDF browser is password-protected. Interest readers can get proper credentials at <http://wiki.larkc.eu/LarkcProject/WP6/Private/AmaSparqlPwd> or ask for them via email to [daniele.dellaglio@cefriel.it](mailto:daniele.dellaglio@cefriel.it).



## 2.3 Term of comparison

In order to get valuations from the LarKC platform in the Urban Computing scenario we decided to design and develop at the same time of the Urban Baby LarKC an application with the role of term of comparison. The main requirement that we fixed in the design of this application is that its behaviour should be the most similar to the Urban Baby LarKC one:

- **DATA SETS:** the term of comparison considers the same data sets used by the Urban Baby LarKC and access the data in the same way (if the Urban Baby LarKC accesses a remote SPARQL endpoint, the application will get the data querying the same service);
- **DATA FLOW:** the term of comparison manages the data flows like the pipeline executed on LarKC (if the Urban Baby LarKC gets the data from two sources, it makes a merge and then it start a reasoning operation on the union dataset then the term of comparison will do the same);
- **ALGORITHMS:** the term of comparison applies the same reasoning algorithms of the Urban Baby LarKC (the code is the same as much as possible).

Developing the term of comparison with the same LarKC methodology we can obtain some indicators about the LarKC platform and how much it contributes on the general application performance and the overhead it may impose.

As we will explain in the section 2.5, there are some pipelines that can be executed on the Urban Baby LarKC. It means that for each of these pipelines an equivalent version of the term of comparison has been realized.

## 2.4 Urban Baby LarKC plug-ins

In order to realize functionalities for the Urban Baby LarKC scenario we designed different plug-ins that we will present in this section (classified per type, according to the five main plug-in types defined so far within the project). These plug-ins has been developed mainly by WP6 partners in collaboration with WP5 ones.

- **IDENTIFIERS:** their main objective is the retrieval of the required information. Actually we have designed four plug-ins of this kind:
  - **RemoteGraphLoaderIdentifier:** it loads triples contained in a known RDF file located on the Web;
  - **SparqlEndpointIdentifier:** it accesses a remote SPARQL endpoint and submits a CONSTRUCT query; it returns the found RDF triples. This plug-in is under development and it has not been completed yet;
  - **GeoLocationIdentifier:** the purpose of this plug-in is to execute geo-based queries. In order to solve this task it uses some of the AllegroGraph geo-spatial features, submitting queries to a remote AllegroGraph server. Received coordinates of a point and a range, this plug-in finds RDF triples indexed by AllegroGraph on the circular area defined by the input range and the input point as center;



- **MixedStrategyIdentifier**: this IDENTIFIER plug-in realizes a more complex strategy to retrieve the data: it looks for the RDF triples defining the areas around the start and goal points involved in the query (using AllegroGraph features) and the main roads that connects them (we will discuss this strategy in more detail in Section 2.5).
- **TRANSFORMER**: we developed a TRANSFORMER plug-in, **SparqlToGeoQueryTransformer**, in order to parse the LarKC input SPARQL query and to extract relevant information from it. The plug-in processes the query and finds the input for the GeoLocationIdentifier and the MixedStrategyIdentifier (centers and ranges).
- **REASONER**: we designed a REASONER plug-in, called PathFinderReasoner, to solve the task of finding the most desirable path. We are developing two different version of this plug-in, to try and to compare different approaches:
  - **OpResPathFinderReasoner**: it is based on the Operational Research: taking advantage of structures and methods of this discipline the plug-in converts the input RDF triples in a suitable graph to apply Dijkstra algorithm on it;
  - **ObBasedPathFinderReasoner**: it is based on Ontobroker from Ontoprise: taking advantage of an existing commercial products functionality, this plug-in converts the input RDF triples in a F-Logic based facts to apply Bellman-Ford algorithm on it;
- **SELECTER**: after data is identified (by IDENTIFY plug-in), in order to better fit data to our needs, a SELECTER is required; we picked an existing plug-in, **GrowingDataSetSelector**, developed by WP5 partners (more information available at <http://wiki.larkc.eu/LarkcPlugins>);
- **DECIDER**: the DECIDER plug-in that we realized, **UrbanBabyLarkcDecider**, builds the pipeline and starts the suitable plug-ins in order to execute it. This plug-in has been customized to execute the different pipelines that we prepared.

These plug-ins are the basic blocks on which first pipelines related to the Urban Computing scenario are built.

## 2.5 Urban Baby LarKC pipelines

In previous sections we mentioned that there are different pipelines that can be built and executed. It is useful in order to try and to compare different algorithms. We planned to improve increasingly the complexity of the Urban Baby LarKC building pipelines more and more complex. Actually we developed three pipeline structures on which plug-ins presented on Section 2.4 could be inserted. All the pipelines try to solve the same problem: given a start and a goal points they compute the most desirable path. It is worth noting that we did not invent yet another route planning approach, on the contrary, we took into consideration “the very naive implementation of Hierarchical Approaches to route planning” (see introduction to Chapter 3 of [6]). More details are offered in Section 2.5.2.

In the following we shortly present the pipelines explaining their main features.

### 2.5.1 First pipeline

Figure 2.2 shows the first pipeline structure we designed. It is the the simplest one: it is composed by an IDENTIFIER, a SELECTER and a REASONER.

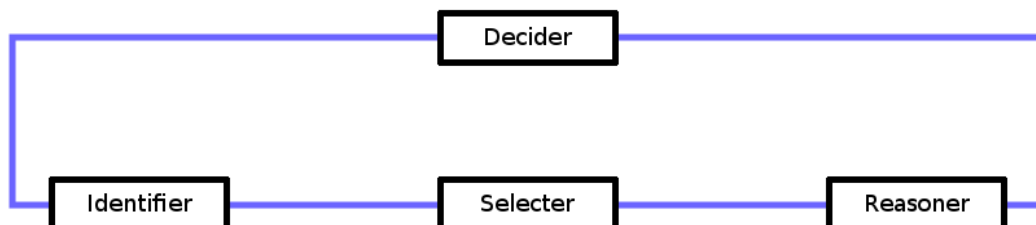


Figure 2.2: First Urban Baby LarKC pipeline

As we explained in the previous section, as SELECTER the GrowingDataSetSelector is used and the REASONER can be one of the two implementations of PathFinderReasoner. There are two IDENTIFIER plug-ins that could work in this pipeline: RemoteGraphLoaderIdentifier and SparqlEndpointIdentifier; their task is to retrieve the RDF triples that describe the area of the city on which the path should be processed.

The pipeline that we build with the RemoteGraphLoaderIdentifier looks for the most desirable path on a fixed small area (we considered the historical center of Milano, an area with a range of about one kilometer), because the loading of the whole graph of Milano requires too much time to execute the pipeline (some minutes). So we have to assume that the start and the goal node are in the considered area, otherwise the path will not be found.

In order to relax the constraint of the fixed area the SparqlEndpointIdentifier can replace the RemoteGraphLoaderIdentifier, enabling the selection of portions of the Milano graph based on the properties that describes roads and points (for example the capacity of the streets or the jurisdiction where the roads are located). Different queries could be submitted to the SPARQL endpoint: depending on the start and goal node the query that retrieve the best portion of the Milano graph changes. When this plug-in is ready we will go into more depth on this facet.

### 2.5.2 Second pipeline

The pipeline structure represented in Figure 2.3 is designed to allow the execution of alternative pipelines where the input LarKC query affects the data retrieval process. The task of extracting relevant data from the query is assigned to the SparqlToGeoQueryTransformer; as in the first pipeline structure the SELECTER is the GrowingDataSelector and the REASONER can be one of the versions of PathFinderReasoner. The possible IDENTIFIER plug-ins that can be involved in this pipeline are the GeoLocationIdentifier and the MixedStrategyIdentifier.

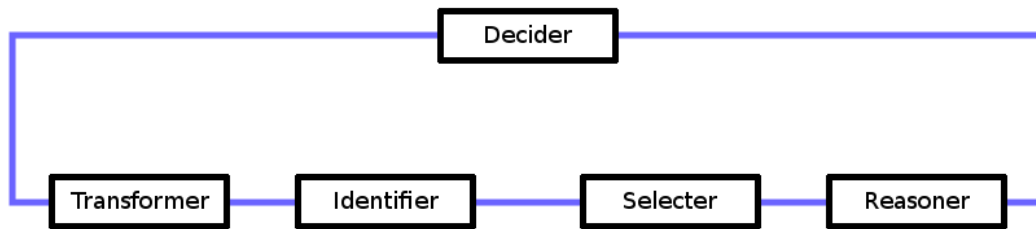


Figure 2.3: Second Urban Baby LarKC pipeline

In Section 2.4 we described the GeoLocationIdentifier explaining that it allows to submit to a remote AllegroGraph service a query extended with geo-spatial features supported by this application. The TRANSFORMER before it extracts the center coordinates and the range of the area that should be considered and provide them to the GeoLocationIdentifier. When this plug-in is used in this pipeline the area to consider in order to find the desirable path between the start and the goal node can be the one with the middle point between the two nodes and as diameter the distance from the start to the node slightly increased. The performance of this plug-in is highly related to the diameter of area: this policy does not work if the distance between the two points is too much (more than some kilometers).

Summarizing what we described until now, the selection strategies of the area where look for the desirable path are simple and they does not work when the distance between the start and goal node are too high. In order to try to solve this problem we tried to develop a more complex approach to select useful portions of Milano graph. According to [5], “algorithms for route planning in transportation networks have recently undergone a rapid development, leading to methods that are up to one million times faster than Dijkstra’s algorithm”; those methods improve performances by a smarter selection of the data subset on which applying standard algorithms; [4] can be considered as the reference implementation of the Hierarchical Approaches to route planning.



Figure 2.4: The complex selection strategy

The strategy that we designed for the Urban Baby LarKC is represented in Figure 2.4. The two markers represent the start and the goal nodes; the idea is to select:

- a subset of roads that covers a big area of the city – we choose to select the subset of the main roads (the roads marked with the thick line);
- the roads around the starting and goal points – the streets inside the circles around the marker.

Merging these three subsets, a graph where start node is connected to the goal one is obtained, so its possible to compute a path.

A first attempt to realize a plug-in that follows this strategy for the Urban Baby LarKC is MixedStrategyIdentifier: we mixed the code of RemoteGraphLoaderIdentifier (to select the main roads) and the one of the GeoLocationIdentifier (to select the roads around a point). This plug-in can replace the GeoLocationIdentifier in the second pipeline structure, in fact it requires some information that is provided by the SparqlToGeoQueryTransformer.

### 2.5.3 Third pipeline

The MixedStrategyIdentifier realizes the complex strategy that we’ve just described above, but it is not a good solution: LarKC is a platform that supports plug-ins and the creation of pipelines where they can be combined, so we can build a pipeline where our strategy is realized with GeoLocationIdentifier, RemoteGraphLoaderIdentifier and the SparqlEndpointIdentifier. Figure 2.5 represents a pipeline structure that can realize this approach.

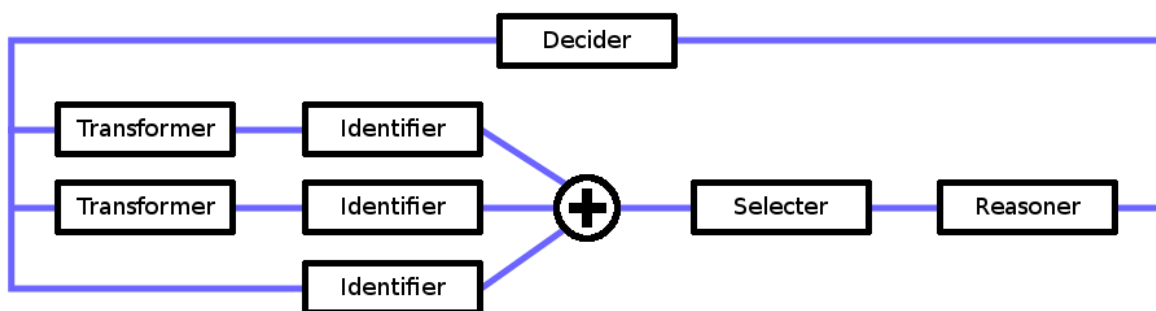


Figure 2.5: Third Urban Baby LarKC pipeline

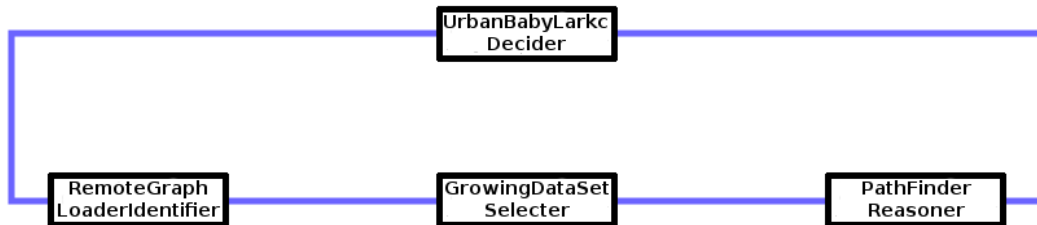
The two couples TRANSFORMER–IDENTIFIER are composed by a SparqlToGeoQueryTransformer and a GeoLocationIdentifier that find the graph around the start and the goal nodes; the third IDENTIFIER plug-in is a RemoteGraphLoaderIdentifier and it loads the RDF model describing the main roads (in future the SparqlToGeoQueryTransformer could replace this plug-in). As the previous pipelines the SELECTER plug-in is the GrowingDataSetSelector and the REASONER is one of the two PathFinderReasoner. A difference from the first two pipeline structures is that in this case a merge is required, so the Data Layer should offer this feature.



### 2.5.4 Recap

Summarizing the actual realizations and implementations of the various plug-ins, as described in the previous sections, we list the pipelines we designed and their current status.

#### Pipeline 1a

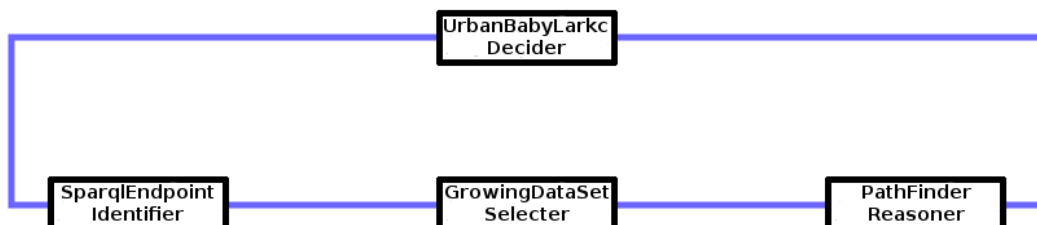


DESCRIPTION: loads the historical center of Milano and find a desirable path between a start and a goal node (inside the area)

STATUS: Working

DECIDER	UrbanBabyLarkcDecider
IDENTIFIER	RemoteGraphLoaderIdentifier
SELECTER	GrowingDatasetSelector
REASONER	PathFinderReasoner

#### Pipeline 1b

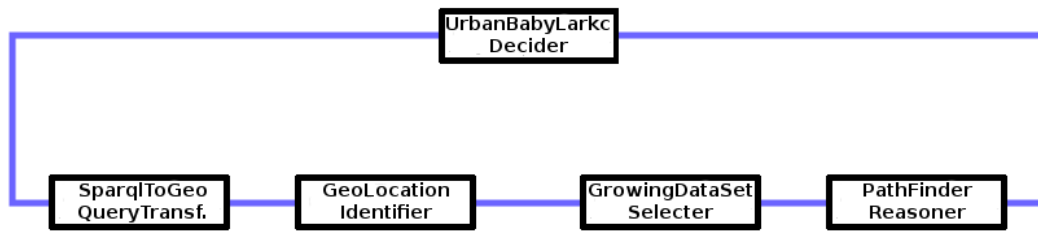


DESCRIPTION: loads a portion of the Milano graph selected with a standard SPARQL query (for example the roads with an high capacity) and find a desirable path between a start and a goal node (located on roads selected by the query)

STATUS: Not working (the SparqlEndpointIdentifier is under development)

DECIDER	UrbanBabyLarkcDecider
IDENTIFIER	SparqlEndpointIdentifier
SELECTER	GrowingDatasetSelector
REASONER	PathFinderReasoner

### Pipeline 2a

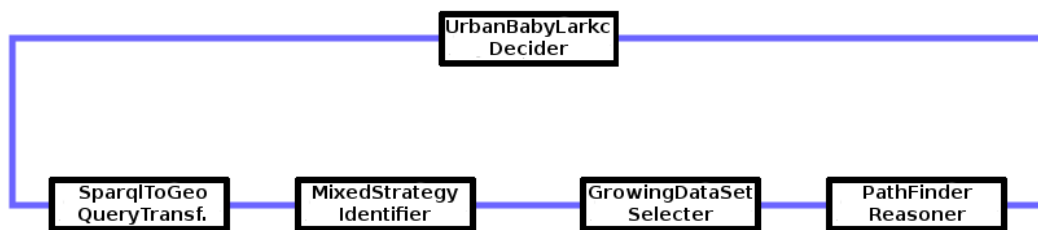


DESCRIPTION: loads the graph of the area containing both the start and the end points and find a path between them (it works only if the two points are near, or it requires too much time to get the result)

STATUS: Working

DECIDER	UrbanBabyLarkcDecider
TRANSFORMER	SparqlToGeoQueryTransformer
IDENTIFIER	GeoLocationIdentifier
SELECTER	GrowingDatasetSelector
REASONER	PathFinderReasoner

### Pipeline 2b

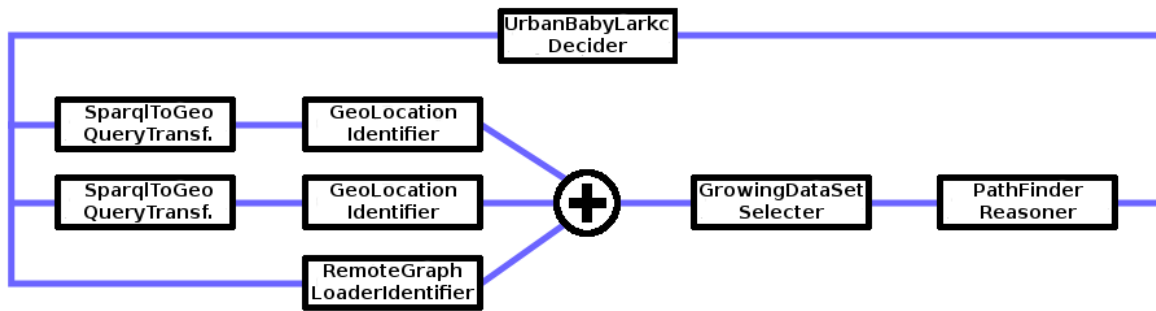


DESCRIPTION: loads the main roads and the graph around the start and the goal nodes and find a path between them

STATUS: Working

DECIDER	UrbanBabyLarkcDecider
TRANSFORMER	SparqlToGeoQueryTransformer
IDENTIFIER	MixedStrategyIdentifier
SELECTER	GrowingDatasetSelector
REASONER	PathFinderReasoner

### Pipeline 3a

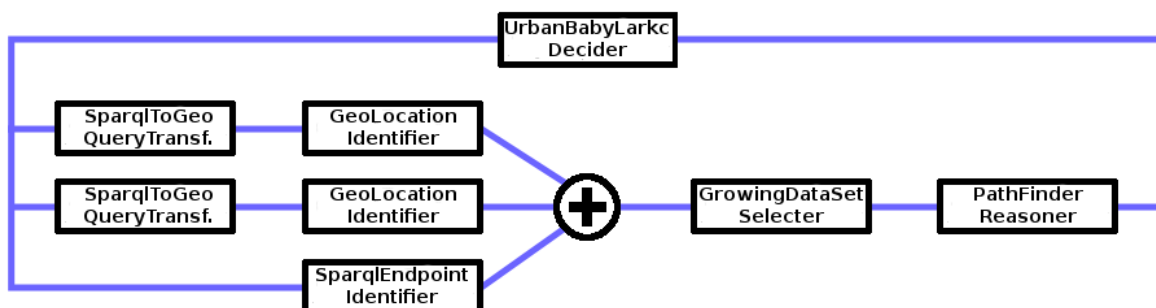


DESCRIPTION: loads the main roads and the graph around the start and the goal nodes and find a path between them (like pipeline 2b but this pipeline is more complex and realized with simpler plug-ins)

STATUS: Not working (the merge is not supported by the platform)

D	UrbanBabyLarkcDecider	
T	SparqlToGeoQueryTransformer (x2)	–
I	GeoLocationIdentifier (x2)	RemoteGraphLoaderIdentifier
S	GrowingDatasetSelector	
R	PathFinderReasoner	

### Pipeline 3b



DESCRIPTION: loads the main roads and the two graphs of areas around the start and the goal nodes and find a path between them (like pipeline 2b and 3a but the main roads are loaded with a SPARQL query)

STATUS: Not working (the SparqlEndpointIdentifier is under development and the merge is not supported by the platform)

D	UrbanBabyLarkcDecider	
T	SparqlToGeoQueryTransformer (x2)	–
I	GeoLocationIdentifier (x2)	SparqlEndpointIdentifier
S	GrowingDatasetSelector	
R	PathFinderReasoner	

## 2.6 Urban Baby LarKC client

While we were developing the pipelines for the Urban Baby LarKC we started the realization of a front–end for the Urban Baby LarKC. It is a Web application offering:

- a simple interface in order to generate in a simple way requests for the Urban Baby LarKC and to show the response in a suitable format (the graphical representation – the path line drawn in the map);
- an interface to observe the behaviour of the execution (the textual representation).

The Urban Baby LarKC Client is actually under development. The graphical representation is almost completed and it was designed to be easily extended in order to support the future development of the Urban Baby LarKC. It is an application developed on the top of Google Maps and actually it allows to select the start and the goal node and to submit the query to the platform (on Figure 2.6).

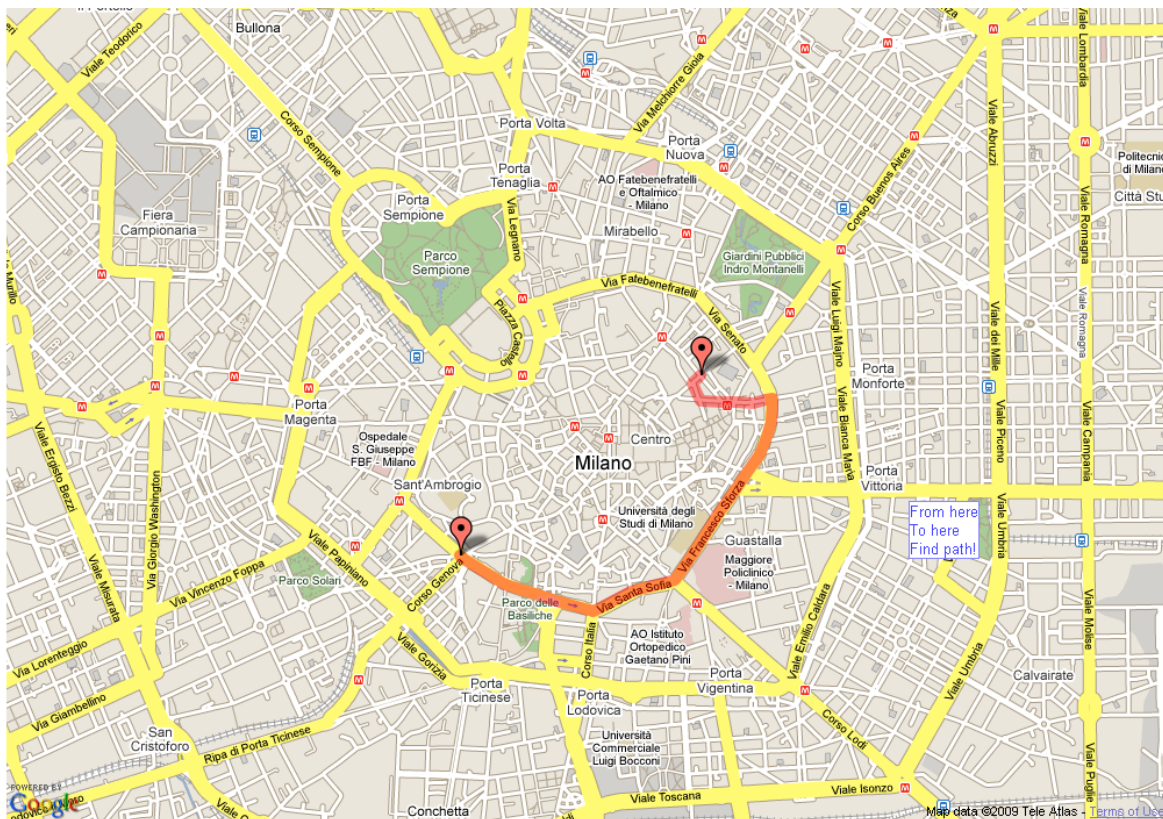


Figure 2.6: Urban Baby LarKC Client

In parallel we are also developing an offline version of the graphical representation: in fact in order to execute the application a Web connection is required to use Google Maps API. Providing an offline version of the Urban Baby LarKC Client can be useful to test the platform when the Web connection is not available.

The textual representation is designed for developers and for users that want to supervise the execution: this part of the interface shows the generated SPARQL query submitted to the LarKC platform, the answer (for example in XML format) and other related information as like the state of the connection between LarKC and the Client and so on.



### 3. Lessons Learned from the Early Adoption of LarKC

In our experience of building the Urban Baby LarKC, we encountered several issues related to the early adoption of the LarKC platform. Some of those issues are strictly associated to the process development of the platform itself and therefore will be solved while the development proceeds and the code becomes more mature; for this reason, those issues are not reported here.

However, trying to apply LarKC technology to a concrete use case, we met some questions and some difficulties that could be interesting to address in the future releases of the LarKC platform. We already discussed some of those issues with the platform developers (WP5), therefore some of the points are already included in the forthcoming updates of the platform components.

Hereafter, we provide a short discussion on those points, adding – whenever feasible and possible – examples from our Urban Baby LarKC scenarios. Those remarks can be considered like a sort of new requirements or feature requests for LarKC.

#### Plug-in Configuration

While building the Urban Baby LarKC pipeline, we developed and inserted in the pipeline some plug-ins to make it realize our scenario.

For example, as discussed in Section 2.4, we developed a `GeoLocationIdentifier` which can be invoked to get the triples within a specific geographic area; in order to do this, the plug-in connects to a remote server where this information is stored and indexed and uses specific address and credentials to query that server.

This kind of data for the connection can be considered as **configuration parameters of the plug-in**; the LarKC platform therefore *should* offer a more sophisticated support to express the plug-in configuration.

#### Merge Operator

The last pipeline structure we described in Section 2.5 (cf. Figure 2.5) envisioned the identification of different portions of the data extracted by different IDENTIFIER plug-ins (one that selects the graph containing all the main roads of Milano, one that selects a graph containing all the roads around the starting point and one the selects the graph containing all the roads around the arrival point). After this identification, therefore, there would be three separate RDF graphs containing parts of the data necessary to be reasoned upon to get the most desirable path between two points.

In order to do this, the pipeline should be able to **execute in parallel** some of the plug-ins and the **merge their partial results** before continuing with the rest of the execution; the LarKC platform therefore *should* offer a merge operator to be applied whenever needed to the plug-ins included in a pipeline.

#### Split Operator

In a similar way, a scenario could include the need for the pipeline to be **split into parallel branches** for a part of its execution (cf. Figure 2.5).

A meaningful scenario highlighting this need should be the previous one – with multiple IDENTIFIER plug-ins run in parallel – or a similar one in which the



pipeline passes the intermediate results to several REASONER plug-ins that are executed in parallel. This can happen, for example, when different kinds of reasoning are needed either to infer different results on the same portion of data or to be applied to different datasets. For example, different reasoning strategies can be applied to find the most desirable path between two points on the Milano graph, in order to take into account different preferences in this search.

In order to do this, the pipeline should be able to **split its execution** into parallel branches; the LarKC platform therefore *should* offer a split operator for the pipelines.

### Pipeline Fragments

In the same way as described before for merging the results of parallel IDENTIFIER plug-ins and for splitting the execution into parallel REASONER plug-ins, there could be the need for merging/splitting entire portions of pipelines or even entire pipelines.

What is needed is a concept of **pipeline fragment** that are formed by a combination of several plug-ins and to which some operators can be applied, like the merge and the split operators. This concept is somehow similar to the idea of process fragment in workflow systems. Please, note that this idea of “fragment” could also be used to “wrap” a complete pipeline as a single “component” in another pipeline. For example, the whole path finding pipeline (described in Section 2.5) can be considered a fragment of the Extension 1 described in Chapter 4.

In our opinion therefore the LarKC platform *should* offer this abstraction of pipeline fragment.

### Pipeline Loops

Another issue that we would like to raise regards the possibility to insert **loops in the pipelines** execution.

In some cases, the loop is directly needed in the pipeline *at design time* – either because a specific portion of the pipeline has to be executed more than one time or to add the so-called anytime behavior that repeats the whole pipeline until the (runtime) decision to stop the execution.

However, while building the Urban Baby LarKC, we met the need for having a loop within the pipeline *at runtime*. In the scenario explained in Section 2.5 and referenced in the “Merge Operator” lessons learned, three IDENTIFIER plug-ins are executed to select three portions of the Milano roads graph (around the starting and arrival point and all the main roads); after this a REASONER plug-in should reason upon the union of those three graphs in order to discover the most desirable path.

In order to do this, the RDF graph merging the three portions must represent a connected graph of roads, otherwise no path can be computed between the starting and the arrival points. Therefore, before passing the data to the REASONER plug-in to compute the most desirable path, a check should be performed on the merged graph to assure the connectivity of the roads graph.



Another example could be the Extension 1 we describe in Chapter 4, which requires to repeat the whole path finding part of the pipeline for every result of the initial search for interesting destinations.

In order to do this, the pipeline should be allowed to **have loops** in a part of its execution and the condition to enter the loop should be evaluated at runtime by the DECIDER plug-in on the basis of the partial results of the execution; the LarKC platform therefore *should* offer the possibility to have loops in the pipelines.

### Data Caching/Persistence and Concurrency

The LarKC platform already offers the possibility to **cache** data; for example, a DECIDER plug-in can keep the data it receives from other plug-ins during the execution of the pipeline. However, in this way, the data is persistently stored until a DECIDER plug-in explicitly delete the data from the cache. The risk, however, is that, if the cache is not emptied, the execution of the same pipeline can give different results depending on the data that is stored in the cache.

For those reasons, a richer data caching behavior could be envisioned; moreover, the **persistence** of data in the so-called Data Layer should be correctly planned and managed, in order to avoid undesired behavior and results. This has particular relevance every time there is a **concurrent access** of multiple components to the shared data: as explained also before, there is a number of circumstances in which different plug-ins should access and operate on the same bunch of data (e.g. when different REASONER plug-ins try to infer knowledge from a dataset by applying different reasoning strategies).

Therefore, we believe that the LarKC platform *should* take care of concurrency, data persistence and data caching.

### Support for plug-in development and management

Whenever a new plug-in is to be designed and developed, some guidelines and “rules” should be followed to correctly realize it and make it work in conjunction with the LarKC platform. For example, every plug-in must obviously be compliant with the API of its generic type.

However, every programmer knows that, when developing a piece of code that must work with a platform, a lot of problems arise for conflicts between the plug-in and the platform. To prevent this, the platform should offer some supporting facilities to ease the development of plug-ins, thus increasing the chance of community contribution and platform extension.

An example of support would be the possibility to let the plug-in developer add some **plug-in-specific files** that are needed for the plug-in correct functioning (something similar to what happens in common Web applications, like servlet containers, with the WEB-INF folder) and that should not be mixed with the platform-specific files and folders.

Another issue that is very relevant for plug-in development is the correct **management of libraries**. It is quite frequent, in fact, that plug-ins adopts or are based on third-party libraries (either open source or even proprietary). Therefore, the possibility exists that different plug-ins use different versions of the same libraries or that incompatibilities arise between libraries.



To this end, the platform *should* arrange separate and independent execution environments for each single plug-in; for example, this result can be achieved by operating at the class loading level. In this way, each plug-in can be executed without interfering with other plug-ins or with the platform.

### Exception Handling and Gentle Recovery

Due to the current status of the platform (“in-progress”), during the coding and execution of the pipelines of the Urban Baby LarKC described in Section 2.5, we faced the problem of understanding where a problem of execution was initiated and how to solve it. We met several times the situation in which the execution of a pipeline encountered some problems and was blocked without any evident sign or signal for the user; moreover, the platform kept working while the client application was not notified of anything. Finally, the recovery of the system after a problem was not easy because the “system” state was not predictable and could interfere with the following (re-)execution of pipelines.

For those reasons, we believe that it is of great importance that the LarKC platform *should* support the correct setup of the **exception handling** and *should* provide a standard way to restore the system after a problem occurred and, if not possible, a way to notify the error to the client.



## 4. Conclusions and next steps

In this deliverable we did not simply present the “Urban Computing environment specification”, i.e. the *design* of a Urban Computing application, but we reported about the realization activities already performed towards the so-called Urban Baby LarKC, the *implementation* of an application that makes use of the LarKC platform and of some plug-ins to concretize some Urban Computing scenario.

This is of course only the first step of our road-map towards a fully fledged Urban Computing application developed with LarKC technologies and tools. As long as the LarKC platform becomes more mature and stable and offers more functionalities – among which the ones we suggested in Chapter 3 –, we will be able to make concrete more complex Urban Computing scenarios, by reusing or implementing plug-ins and by defining and executing new pipelines over the LarKC platform. As explained in Chapter 1, in fact, we want to realize a richer use case that involves more data sources and that solicits and stresses more the LarKC platform, demonstrating, in the meantime, its advantages with regards to existing technologies.

We already envision some possible extension to the scenario already implemented via the pipelines described in Section 2.5. Both extensions involve new data sources to be taken into account, but their requirements and their consequences on the pipeline are as follows:

**Extension 1** the *destination* of the path is *dynamically chosen*:

The user does not know in advance what the destination is of his route, but he expresses some requests or ambitions and the destination is selected on the basis of those preferences; for example, the user says that he would like to go and visit some interesting monuments or venues of a city, or that he would like to attend some music concerts or cultural event that night, or that he would like to meet some of his friends that happen to be in the same city.

The impact on the pipeline is that, first of all, a query should be routed to an appropriate data source (an archive of points of interests, a source of events schedule, a localization systems for a social network) and should select some possible destinations; then, for each destination, a suitable strategy to find the most desirable path (for example, among the ones described in Section 2.5).

**Extension 2** the path finding takes into account the *traffic conditions*:

The user would like the path finding to take into account the traffic conditions on the specific day and at the specific time in which he will have to move.

The impact in this case is that, when calculating the most desirable path, the “weight” of the street links should not be simply the length of the roads, but a number based on the traffic prediction. To this end, a new plug-in should be design and developed which produces traffic predictions on the basis of historical/statistical data; its forecasts should therefore become an additional input for the REASONER plug-in to calculate the suitable path.

We decided to follow this rapid prototyping approach in order to enable an early problem/bug discovery, to suggest possible improvements or to request new functionalities. We believed that this activity – even though it was not planned in the original Description of Work – demonstrated to be very useful both for the technical work



packages (WP2-WP5) and for WP6 itself, which can better plan the following steps on its road-map to the realization of a complete Urban Computing scenario.

In the future deliverable related to the “Urban Computing environment” (D6.5, D6.8 and D6.10), we will update about the realization of a series of demonstrators of increasing complexity that take advantage of the more and more mature implementation of the LarKC platform and the richer support it will offer for the plug-ins development and the pipelines construction and execution.



## REFERENCES

- [1] Christian Bizer and Richard Cyganiak. D2R-Server - Publishing Relational Databases on the Web as SPARQL-Endpoints. In *Proceedings of the 15th International World Wide Web Conference (WWW2006)*, May 2006.
- [2] Emanuele Della Valle, Irene Celino, Daniele Dell’Aglío, Zhisheng Huang, Kono Kim, Werner Hauptmann, Ralph Grothmann, Yi Huang, and Volker Tresp. D6.1 - Requirements summary and initial data repository, September 2008. Available from: <http://www.larkc.eu/deliverables/>.
- [3] Daniele Dell’Aglío, Emanuele Della Valle, and Irene Celino. D6.4 - 1st periodic report on data and performances, May 2009. Available from: <http://www.larkc.eu/deliverables/>.
- [4] Peter Sanders and Dominik Schultes. Engineering highway hierarchies. In Yossi Azar and Thomas Erlebach, editors, *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*, volume 4168 of *Lecture Notes in Computer Science*, pages 804–816. Springer, 2006.
- [5] Peter Sanders and Dominik Schultes. Engineering fast route planning algorithms. In Camil Demetrescu, editor, *Experimental Algorithms, 6th International Workshop, WEA 2007, Rome, Italy, June 6-8, 2007, Proceedings*, volume 4525 of *Lecture Notes in Computer Science*, pages 23–36. Springer, 2007.
- [6] Dominik Schultes. *Route Planning in Road Networks*. PhD thesis, Universität Fridericiana zu Karlsruhe, February, 2008.
- [7] E. Della Valle, I. Celino, D. Dell’Aglío, K. Kim, Z. Huang, V. Tresp, W. Hauptmann, Y. Huang, and R. Grothmann. Urban Computing: a challenging problem for Semantic Technologies. In *Workshop on New forms of Reasoning for the Semantic Web: scalable, tolerant and dynamic (NEFORS 2008), colocated with the 3rd Asian Semantic Web Conference (ASWC 2008), Bangkok, Thailand, 2008*.
- [8] E. Della Valle, I. Celino, K. Kim, Z. Huang, V. Tresp, W. Hauptmann, and Y. Huang. Challenging the Internet of the Future with Urban Computing. In *First International Workshop on Blending Physical and Digital Spaces on the Internet (OneSpace 2008), colocated with the Future Internet Symposium (FIS 2008), Vienna, Austria, 2008*.