WP4-D2

Intelligent UI Design

Project full title:	Knowledge Driven Data Exploitation
Project acronym:	K-Drive
Grant agreement no.:	286348
Project instrument:	EU FP7/Marie-Curie IAPP/PEOPLE WP 2011
Document type:	D (deliverable)
Nature of document:	S (Specification)
Dissemination level:	PU (public)
Document number:	ISOCO/WP4-D2/D/PU/a1
Responsible editors:	Ronald Denaux, Panos Alexopoulos, Honghan Wu, and Andrew Walker
Reviewers:	??
Contributing participants:	ISOCO, UNIABDN
Contributing workpackages:	WP4
Contractual date of deliverable:	15 December 2013
Actual submission date:	15 December 2013

Abstract
This deliverable presents the UI-design and system-design for the data summarisation and the question answering components of the K-Drive system. The functions of the components are first specified through detailed usage scenarios based on the system and UI requirements (respecitely identified in D1.2 and D4.1). Then, technical specification are given for both the interface and the system. The resulting UI and system specifications are suitable for enabling the implementation of the system.

Keyword List
System Design, User interface Design, Dataset Summarisation, Question Answering

Project funded by the European Commission within the 7th Framework Programme/Marie Curie Industry-Academia Partnerships and Pathways schema/PEOPLE Work Programme 2011.

Intelligent UI Design

Ronald Denaux[1], **Panos Alexopoulos**[1], **Honghan Wu**[2], **and Andrew Walker**[1,2]

[1] iSOCO, Spain
[2] Department of Computing Science, Aberdeen University, UK

15 December 2013

Abstract
This deliverable presents the UI-design and system-design for the data summarisation and the question answering components of the K-Drive system. The functions of the components are first specified through detailed usage scenarios based on the system and UI requirements (respecitely identified in D1.2 and D4.1). Then, technical specification are given for both the interface and the system. The resulting UI and system specifications are suitable for enabling the implementation of the system.

Keyword List
System Design, User interface Design, Dataset Summarisation, Question Answering

Contents

1 Introduction

Deliverable D1.1 Alexopoulos et al. (2012) introduced the use-cases for the K-Drive project. In deliverable D4.1 we further identified the need for tools for supporting end-users (and developers) in dataset exploitation tasks such as getting answers for their questions out of existing dataset and being able to get tailored summaries of dataset based on users' requirements and intended usage of datasets. In this deliverable we further develop the ideas for two systems – a question answering system and a goal-driven dataset summarisation system – and provide functional and technical descriptions for these systems in order to facilitate their development.

The rest of this deliverable consists of the functional and technical specifications of both systems. We take a scenario-driven approach for these functional specification, where we identify the typical users who will be using the systems. Based on these user-profiles, we then describe the workflow of the systems and the various screens that users will see in order to interact with the systems. The technical specifications present a high-level architecture and a description of the main components for both systems.

2 Functional Specification

This section describes a usage scenario of the data summarisation and question answering systems by a typical user. The described scenarios capture in detail the user's view of the systems functionalities. The scenarios serve as a specification of the functionalities that the systems should provide, but do not specify the underlying technical details and algorithms which are used by the systems (this is done in Section 3).

2.1 Goal-Driven Dataset Summarisation

2.1.1 Typical User

Cristiano Messi is a (imaginary but) typical user of the goal-driven dataset summarisation application. He is a data engineer working for a company that collects and provides access to multimedia items about sport events, mainly football matches involving the main teams from the european top leagues. Because the multimedia items (e.g. pictures, videos, sound recordings) come from different sources and because the company has been gathering these multimedia items for years, the metadata used to describe the multimedia items is of varying quality. This is a main source of problems for Mr. Messi, since the company's customers are not always happy with the search functionality provided by the company and his bosses are pushing him to come up with a better search functionality.

The main problem is that most of the metadata for the multimedia files is in the form of plain text. However, this text is often not being indexed correctly due to various issues (e.g. players are sometimes only referred to by their number, or by their first or last name and teams are only referred to by their colours or by their hometown).

Mr. Messi's company does not have the resources to manually annotate all the incoming multimedia files to improve this, so Mr. Messi is looking into ways to automate the annotation of their files. He knows, through inspecting the search log files, the main types of queries that customers use to find

multimedia files and their *related aspects* (e.g. search by modality, by player name, by team, by sport-related event depicted). Mr. Messi also has learned, that he needs to have a dataset encoding knowledge about all of these aspects, in order to have good results with automatic annotation of the metadata. However, he does not have any experience (or a budget for) building such datasets. Mr. Messi has therefore decided to use the goal-driven dataset summarisation application to find potential public datasets which may be useful.

2.1.2 Overall Workflow

Figure 1 shows the main screens that end-users of the system see when interacting with the goal-driven dataset summarisation application.

Since it is a web application, users will generally start at the homepage (see Section 2.1.3), where they get an explanation about the application and where they can decide to sign up (Section 2.1.4) or to log into the system.

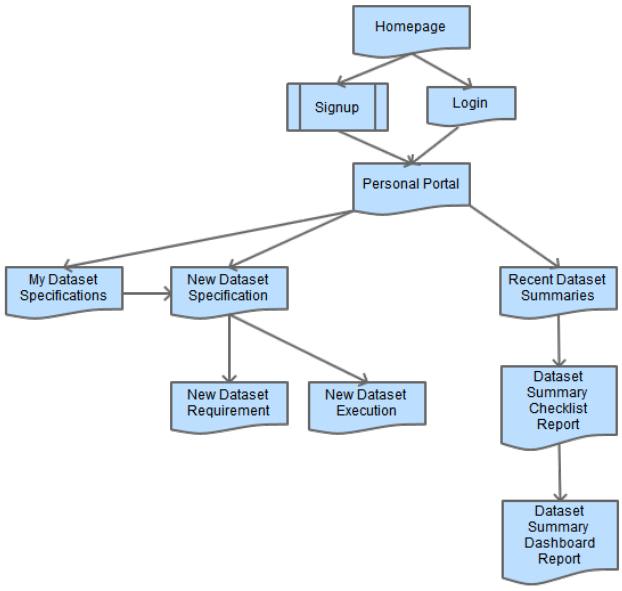

Figure 1: Overall workflow for the Semantic Data Summarization Platform user interface.

Once a user has logged in, they are directed to their personal portal (see Section 2.1.6. The personal portal is the entry point for the two main tasks that users can perform on the goal-driven dataset summarisation platform:

Specifying their dataset requirements From the personal portal users can see and operate their list of dataset specifications (Section 2.1.7).

Viewing dataset summarisations based on their requirements From their personal portal users can also see their recently computed Dataset Summaries (Section 2.1.10) and they can navegate to see more detailed views of those summaries.

2.1.3 Homepage

This is the page that presents the system to the user. It provides an explanation about the main functionalities of the application to help users understand what they can expect from the system and decide whether they want to use the system. A mockup screen for the homepage is shown in figure 2.

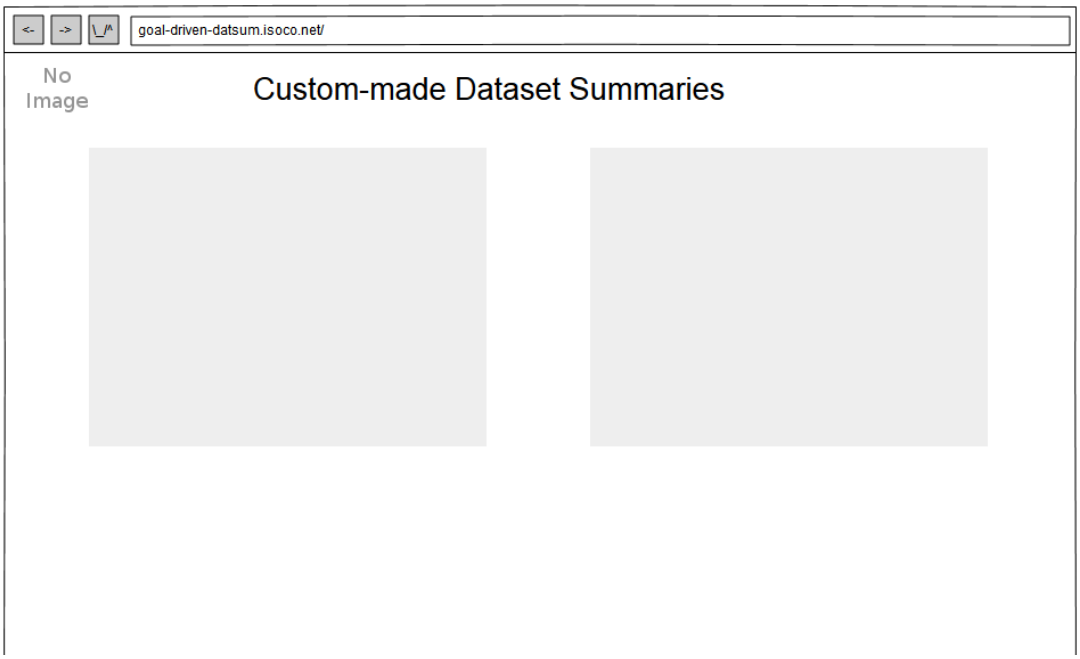

Figure 2: Mockup of the Homepage for goal-driven dataset summarisation.

The homepage also provides links to allow users to sign up (Section 2.1.4) or to log into the system; both of these actions provide access to the personal portal (see Section 2.1.6).

2.1.4 Signup

The sign-up screen allows users to enter their data. The required data from the user is:

username the handle by which the user will be recognised by the system. This handle uniquely identifies a user.

email a valid e-mail address used to send links to retrieve forgotten passwords and (in the future) to send notifications about the system. This address also uniquely identifies a user.

password a secret password that the user will enter to log into the system.

Technical Note

Any additional information is nice to have, but not required. We leave it to the implementation platform to help provide the basic functionalities for sign-up, account creation, password recovery, etc.

2.1.5 Login

Allows users to identify themselves. It requires a combination of the user's username (or the user's email) and their password.

> **Technical Note**
> As with the Signup screen, we rely on the implementation platform to provide basic functionality such as authentication, links to password recovery, etc.

2.1.6 Personal Portal

The personal portal is the entry point for the two main tasks that registered users can perform on the goal-driven dataset summarisation platform:

Specifying their dataset requirements From the personal portal users can see and operate their list of dataset specifications through their list of Dataset Specifications (Section 2.1.7).

Viewing dataset summarisations based on their requirements From their personal portal users can also see their recently computed Dataset Summaries (Section 2.1.10) and they can navegate to see more detailed views of those summaries.

A mockup screen for the personal portal is shown in figure 2.

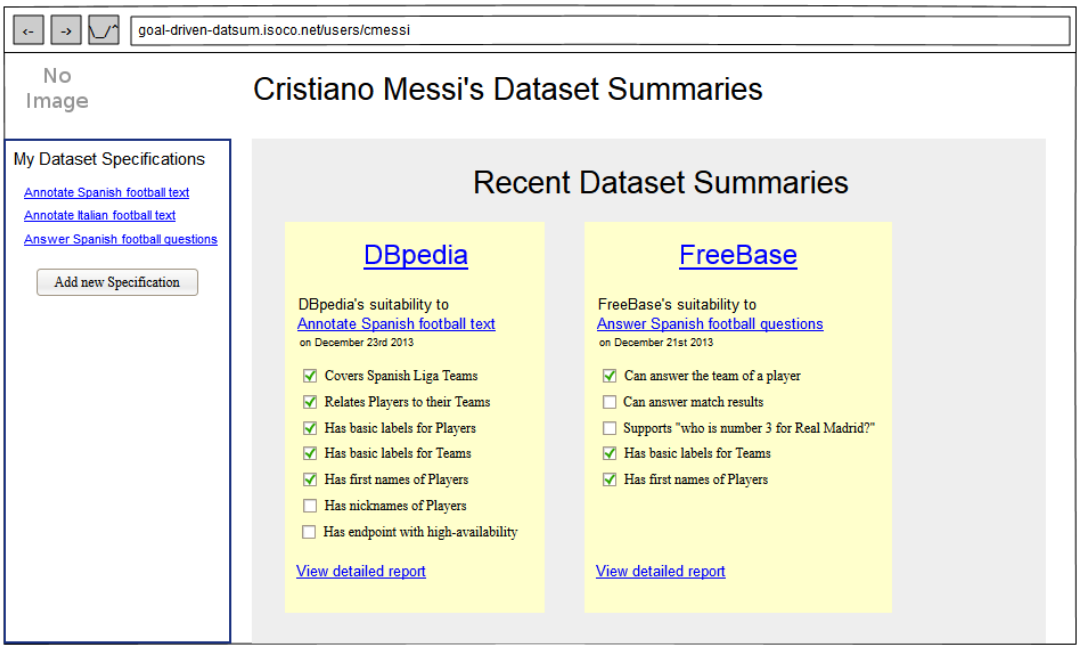

Figure 3: Mockup of the Personal Portal page for goal-driven dataset summarisation.

2.1.7 My Dataset Specifications

The left hand side of the **Personal Portal** displays the user's dataset specifications. See figure 2 for an example list. It consists simply of:

- a list of the dataset specifications, where each item in the list is the dataset specification title. Each item is a hyperlink, so that clicking on the link requests the *Dataset Specification View* screen.
- a button to **add a new** *Dataset Specification*.

2.1.8 New Dataset Specification

This screen allows a user to enter a new dataset specification. It consists of:

- an input box where the user can enter the specification title: a short description of what the user is needs in a dataset.
- an area for adding, removing and editing new dataset requirements.

Figure 4 shows the screen the user sees when they visit the page. The focus of the cursor is on the field to enter the dataset title and the button to add new requirements is disabled. Once the user has entered the title for the specification, the button to add new requirements becomes enabled. When the user clicks on the *Add new Requirement* button, the specification is saved and the user is redirected to the *New Dataset Requirement* screen.

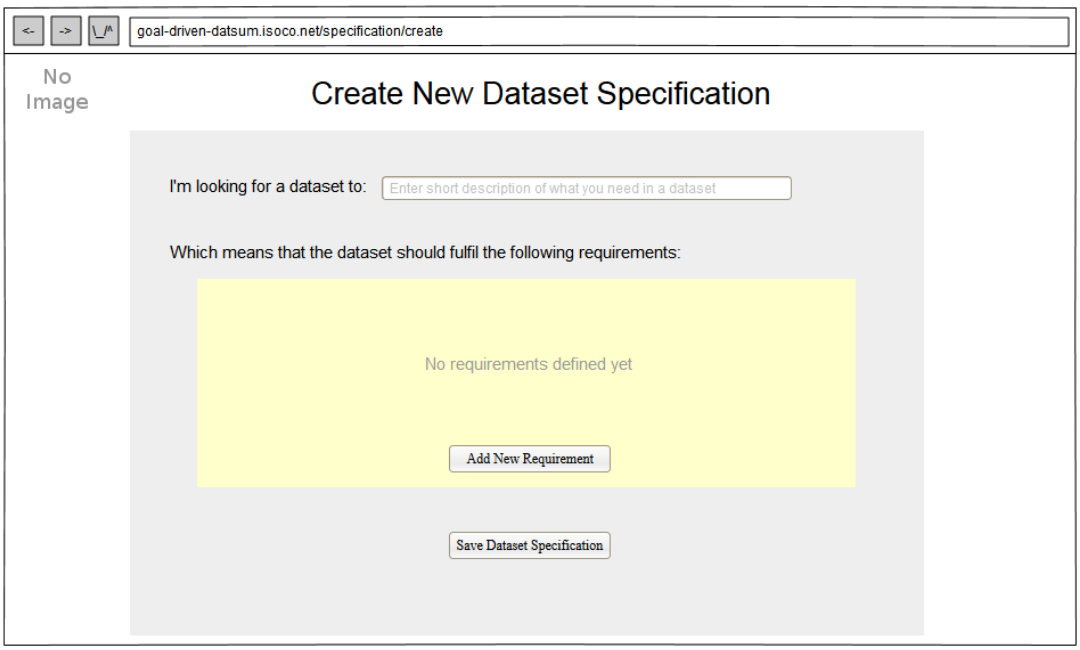

Figure 4: Mockup of the *New Dataset Specification* screen as displayed when user visits ths screen.

2.1.9 New Dataset Requirement

This screen allows a user to enter a new dataset requirement. It first describes that the requirements is part of a dataset specification by linking back to the dataset specification. It then lets the user enter a short description of the requirement being defined. Then, the user needs to select one of the predefined *dataset operations* which will be used to check that a dataset complies with the described requirement.

The list of `dataset operations` will depend on the configuration of the system. For example, an initial version of the system may contain only the following operations: typed entity coverage checker, relationship between typed entities checker, label coverage for typed entities checker, endpoint availability checker. The definition and description of these dataset operations is not in the scope of this deliverable.

A catch-all "dataset operation" will be included, which is not actionable (i.e. it cannot be executed). It's description is simply "another means (not yet supported by the system)". When the user selects this option, the user is presented with a text area where the user can enter free text to describe the property that they want to check on a dataset, but for which no other dataset operation can be used. This serves to collect feedback regarding the types of checks that users want to perform.

2.1.10 Recent Dataset Summaries

The *Recent Dataset Summaries* component is shown as part of the user's *Personal Portal* screen. It displays the *Checklist Report* of available, recent dataset summary executions. Only the n most recent reports are shown, depending on how many are available and how many fit on the screen (where n can be configured by the programmer).

2.1.11 Dataset Summary Checklist Report

The Dataset Summary *Checklist Report* is the shortest type of goal-driven dataset summary. It gives a short summary of how well a specific dataset fulfils the various requirements of a specification. Figure 3 shows two mockups of such a checklist. The report consists of:

- the name of the dataset that was checked with a link to a page relating to the dataset
- a link to the *Dataset Specification* that generated the report
- the date when this report was generated
- the list of *Dataset Requirement*s that were checked. Next to each requirement, the system will display an icon that denotes how well the dataset performs against the requirement. In the first version of the system, we will support only 3 icons: a check denotes that the dataset fulfils the requirement, a cross denotes that the dataset does not fulfil the requirement and a question mark denotes that the requirement could not be checked. In future versions we may introduce new icons to give a finer grained report.
- a link to a more detailed version of the report, which leads the user to the dataset summary *dashboard report*.

2.1.12 Dataset Summary Dashboard Report

Besides the *Checklist Report*, each dataset operation also can be used to generate a detailed report. It is out of the scope of this deliverable to specify how such a detailed report will look like, since we are not

specifying individual dataset operations. However, typically, we envisage dashboard reports to include graphs and/or tables illustrating the results of the dataset operations.

2.1.13 Non-Goals

There are various features that have not been included in the current functional specification, but which have been identified as potential features for future versions of the application.

Alerts & Notifications In the first version of the system, users will have to visit the website regularly in order to check for updates on new summaristion reports.

Trend reports The first version includes summarisation reports as checklists and as dashboards. However, these reports only show the result of one execution of checking dataset requirements agains a particular dataset. In the future it will be useful to provide reports, which summarise the trend of these results over time.

Comprehensive set of dataset operations As explained in Section 2.1.9, a dataset requirement is specified as a dataset operation with specific parameter values. In the first version of the system, we will only implement a couple of such dataset operations to research the viability and usability the platform. In future versions we expect to implement more such dataset operations, which may require advanced functionality for searching and selecting suitable dataset operations for a given requirement. Also, it may be necessary in the future to provide a way of combining dataset operations in a requirement.

Social aspects Although the system allows users to sign-up, users of the system cannot explore the profile of other users or communicate with other users of the system. It may be useful to provide such functionality in the future; for example, to allow more advanced users to help new users to specify their requirements in terms of dataset operations. Another social aspect may involve sharing interesting dataset summarisations via social networks.

Recommendations Besides the social aspects described above, another feature that could be interesting in the future is that of providing recommendations of dataset requirements, dataset operations to check a requirement, users who have similar requirements, datasets which may be worth focusing on, etc.

2.2 Question Answering

2.3 Typical User

The question answering system will cater first and foremost to casual users (users without technical expertise). However, the system will also provide "debugging" information that will be useful for technical users in order to help those users improve their datasets to make question answering easier on those datasets.

2.4 Overall Workflow

The overall workflow for the question answering system is fairly simple. The homepage of the system will encourage users to pose their questions. By default, all questions will be interpreted against all the datasets known to the system, but the users will have the option to specify a dataset to use to answer the question. Once the user has posted their question, the system will try to find answer(s). If answers are found, these are displayed on the *Display Search Result* page. If no answers are found, the user is directed to the *Debug* page. Figure 5

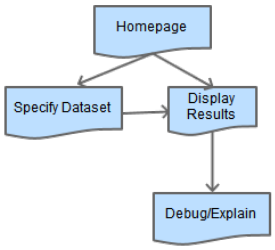

Figure 5: Overall workflow of a user interacting with the Question Answering system.

2.5 Homepage

The homepage is the main screen that the user sees when accessing the system. It contains an input field where users can enter a question for the system to answer. By default, the system will try to determine the best answer by selecting the best dataset from which the answer can be extracted.

The homepage provides a link for *specifying a dataset*. When the user specifies a dataset, the system will only use this dataset to try to answer the question.

Once the user posts a question, the system will try to find an answer and, depending on whether an answer is found or not, the user will be redirected either to the *Display Search Result* or to the *Debug* page.

Figure 6 shows a mockup of the homepage screen.

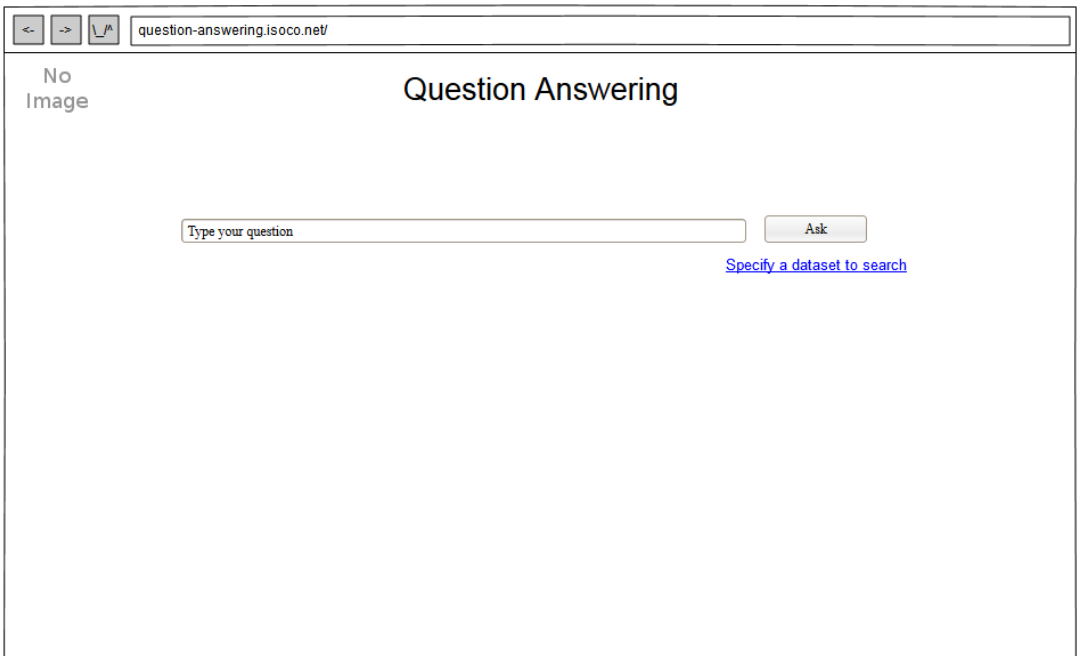

Figure 6: Mockup of the Question Answering *Homepage* screen.

2.6 Search in Specific Dataset

If the user chooses to specify a dataset to use during question interpretation, the user will see a screen similar to the one depicted in Figure **??**. Users will only be able to choose from a pre-defined list of datasets.

2.7 Display Search Result

When a question posed by the user can be interpreted by one of the system's datasets, the system will show the results in a rendering. A typical rendering will be a table showing the results and using a human-readable label, linked to the original resource in the searched dataset. The *Result* screen will show the original posed question as well as the dataset from which the answers were derived.

The *Result* screen will also show a link to the *Debug* page. Note that if the question cannot be answered satisfactorily (i.e. because none of the datasets resulted in an interpretation that could be translated into SPARQL), no *Result* page will be shown, but the user will be redirected to the *Debug* page for the question immediately.

Besides the answers, this screen will also show the search input field, to allow users to enter a new question. Figure 7 shows an example answer screen for the QA system.

2.8 Debug

The system will show the *Debug* screen when:

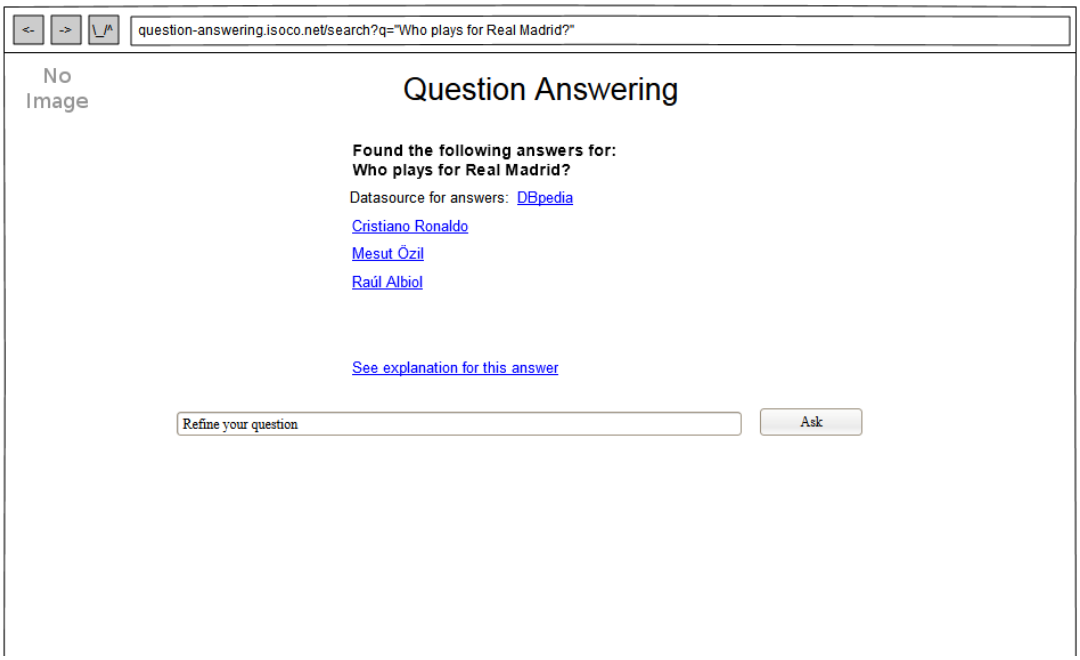

Figure 7: Mockup of the *Answer* screen for Question Answering.

- the system cannot find an interpretation to a question against any of the available dataset (or against the selected dataset)[1].
- the user requests the debugging of an answer.

When no interpretations could be found, the debug screen will show a list of the main problems that occured when interpreting the question against the various datasets. As far as possible, the explanations will avoid overly technical language to make the interpretation understandable to casual users. Figure 8 shows an example of how such a screen may look like.

When a question has been interpreted, the *Debug* screen will also contain a longer explanation about how the original question was interpreted.

Besides the debug explanations, this screen will also show the search input field, to allow users to enter a new question.

3 Technical Specification

3.1 Data Summarisation

3.1.1 System Architecture

In this section we describe the architecture of the dataset summarization system, in terms of the components it comprises and the main functions these serve. The architecture is depicted at figure 9 and

[1] In cases where multiple interpretations are possible, the system will choose the best possible interpretation.

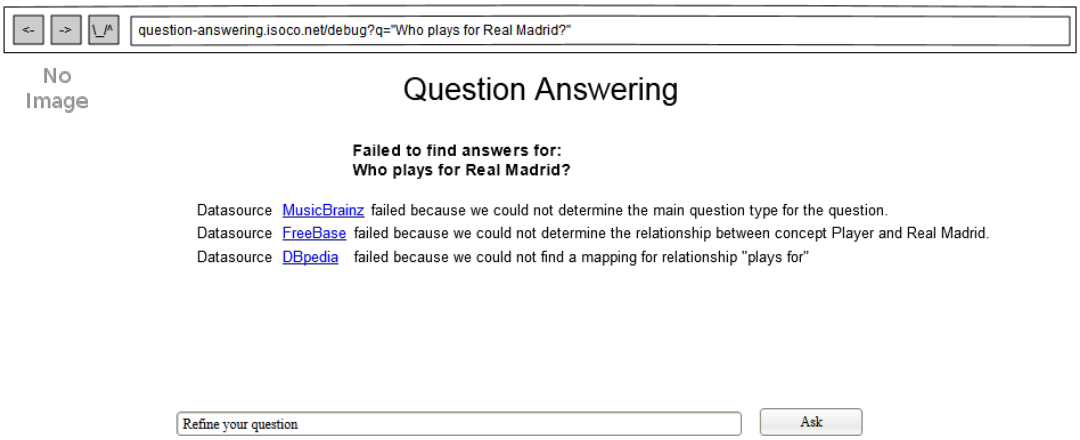

Figure 8: Mockup of the *Debug* screen for Question Answering.

consists of three layers, namely the **data layer** where all the necessary data are stored, the **application layer** that implements the requirement functionalities for the generation and management of summaries and, finally, the presentation layer that implements the systems User Interface.

In more detail, the main components of the system are the following:

RDF Repository This is a repository that stores two types of data, namely i) the semantic datasets that the system is supposed to analyze and produce relevant summaries about and ii) the summarization tasks that are to be applied to the datasets used for the summary generation. For performance and reliability reasons, the datasets are stored locally and updated periodically from their original sources. As for the summarization tasks, these are defined and stored as instances of the ontological schema of figure 10.

Semantic Dataset Manager This is an API for accessing and manipulating, at a low level, the datasets that are to be used within the system and it is primarily used by the Summary Generator component.

Summarization Task Manager This component provides a comprehensive API for accessing and manipulating, at a low level, the summarization tasks that users define and use. It interacts with the RDF repository where the tasks are stored as well as the components that require its information like the Summary Generator and the Summary Visualizer.

Summary Generator This is the core component of the overall system, primarily responsible for executing the user-defined summarization tasks and producing corresponding summaries that are also stored in the system's repository. It implements a predefined set of data operations that the user may define in his/her summarization tasks and executes them when invoked.

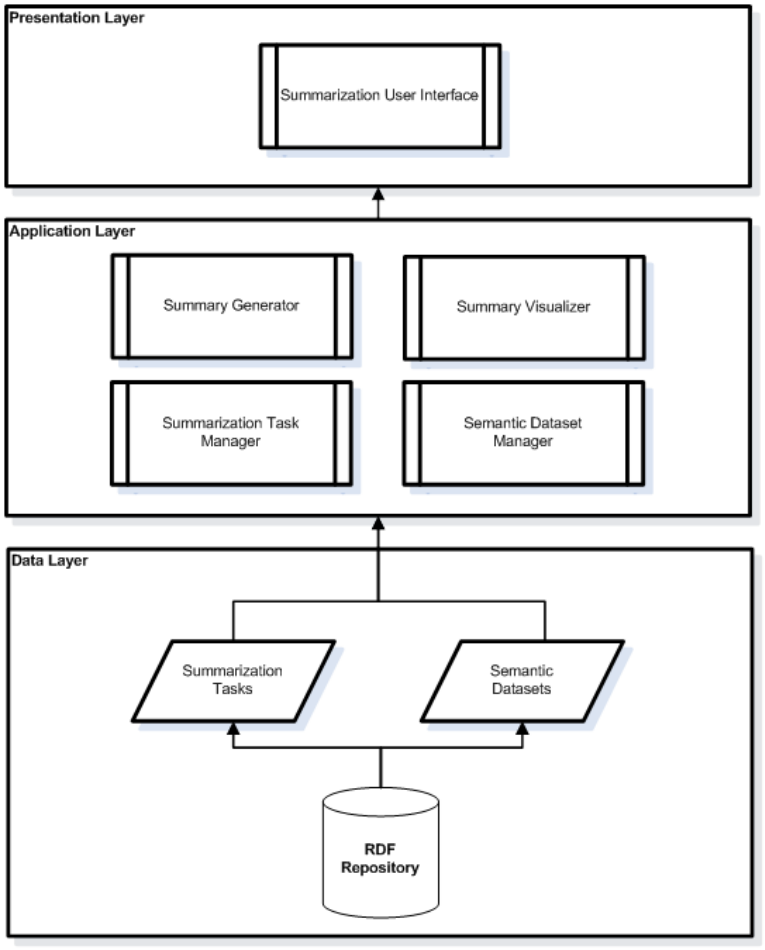

Figure 9: Semantic Data Summarization Platform Architecture

Summary Visualizer This component produces, for a given generated summary, a set of visualization elements that are to be shown to the user at the Presentation layer. These elements are typically predefined and associated to summarization requirements, as shown in figure 10. This association is based on an a priori analysis about what visualization elements best visually communicate the information a requirement generates.

Summarization User Interface This is the main channel through which the system's users are able to define summarization tasks and produce relevant summaries. The task definition is facilitated primarily by means of forms while the produced summaries are visualized according to the dashboard paradigm. Figure 11, illustrates the main elements of the user interface.

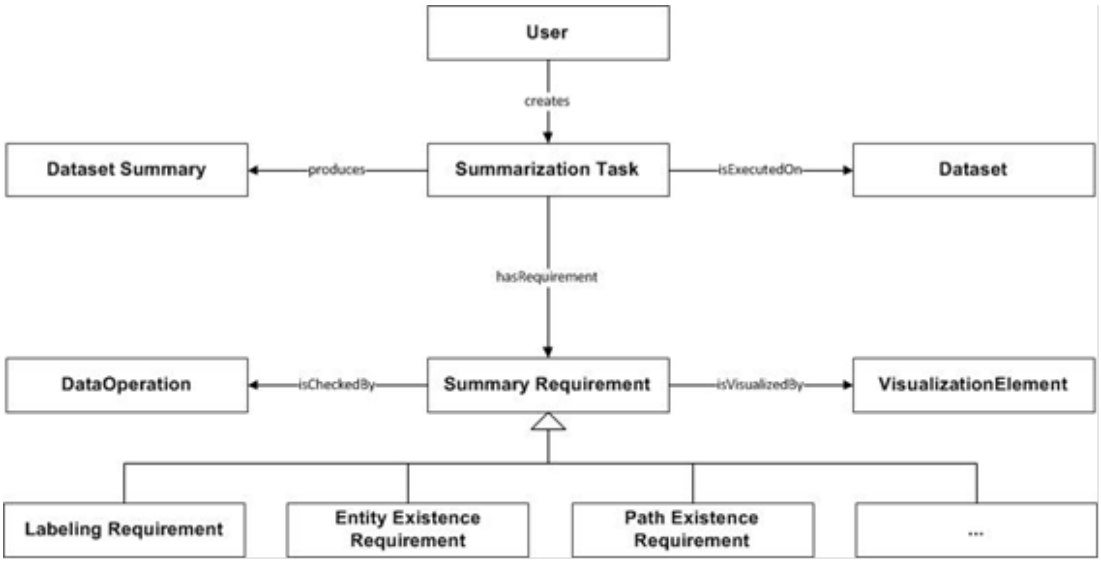

Figure 10: Ontological Schema for Summarization Tasks

3.2 Question Answering

3.2.1 System Architecture

The QA system is a pipeline consisting of three high-level, conceptual, stages. A new natural language question is first *prepared*, for *processing*, before being *formalised* into a language that can be executed against a knowledge-base. *Preparation* is handled by a `Parser` which annotates the question with parts-of-speech and dependency relations. The *processing* stage consists of two components: a `Mapper`, which identifies candidate concepts, relations and entities that terms in the input may be referring to, and a `QuestionAnalyser` which performs auxiliary analysis on the requirements of the question. *Formalisation* also consists of two components. Before the question can be written as a single query, it is necessary to identify precisely how the mappings are related in the data source. This is handled by a `Router`, and the question can then be converted with a `Formaliser`. This sequence is depicted in Figure 12.

Each module has some high-level purpose and responsibilities, where implementations are free to explore the scope of these more specifically.

Parser : A parser must accept a natural language input and provide representations of its grammatical structure in terms of parts-of-speech and dependency relations.

Mapper : A mapper will take the tokens identified by the parser and list possible entities, concepts and/or relations from the data source that may represent them. It must make these mappings available and also record the classes of any instances it maps, i.e. if "Real" were mapped to the football team `dbp:Real_Madrid_C.F.` then it should also record that `dbp:Real_Madrid_C.F.` is an instance of type `dbpo:SoccerClub` (etc).

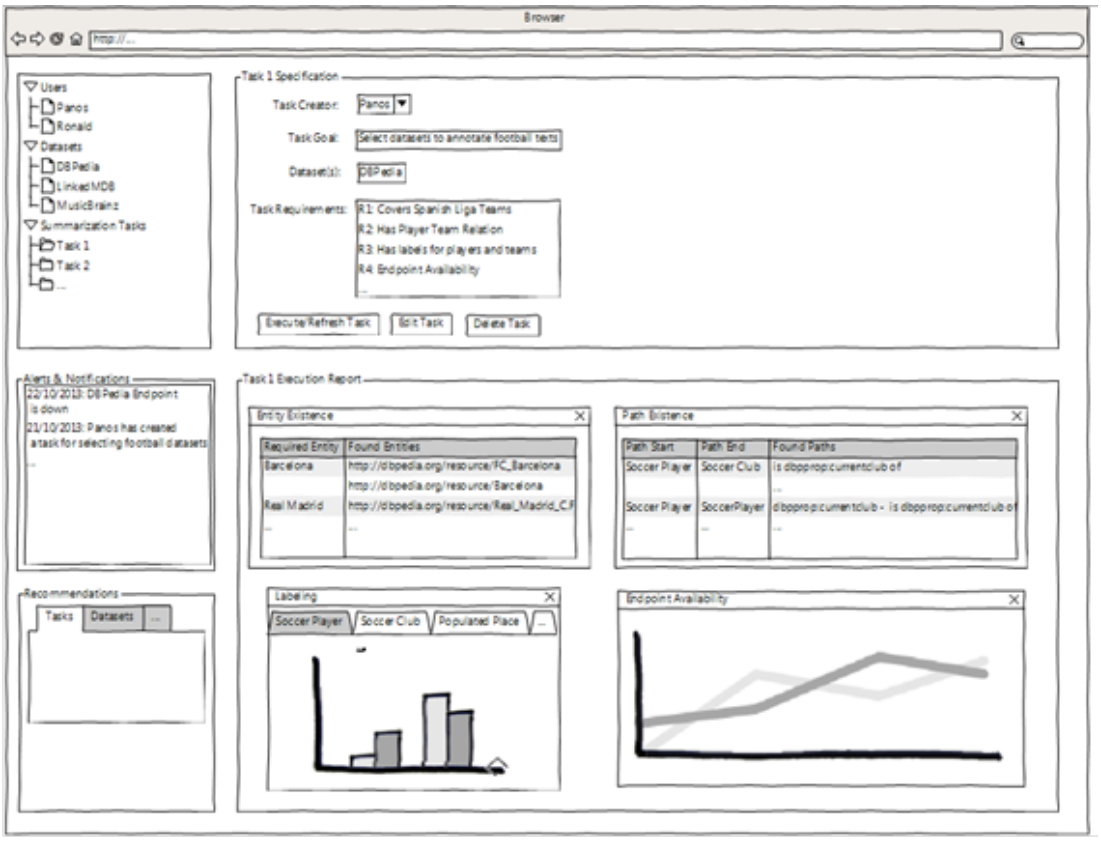

Figure 11: Summarization System User Interface

QuestionAnalyser : A question-analyser provides auxiliary information about the nature and requirements of the question. For instance, it will identify the semantic class of the desired response. In addition, it may perform date-time analysis and handle superlatives.

Router : The router identifies and selects the triple-paths in the data source that connect the mapped elements of the question. For example, in a question "What is the capital city of Spain?", we might find three mappings: capital → `dbpr:capital` (a relation), city → `dbpo:City` (a class), and Spain → `dbp:Spain` (an instance of `dbpo:Country`). The router must then find a way to connect these three mappings.

```
<http://dbpedia.org/resource/Spain>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://dbpedia.org/ontology/Country> .
<http://dbpedia.org/resource/Spain>
<http://dbpedia.org/property/capital>
<http://dbpedia.org/resource/Madrid> .
<http://dbpedia.org/resource/Madrid>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
```

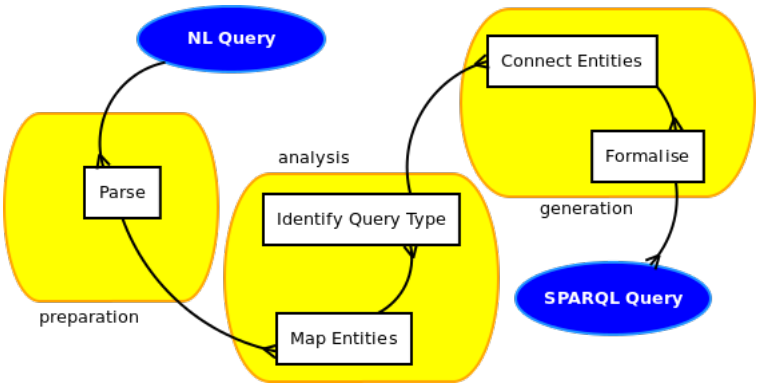

Figure 12: Pipeline of components for Question Answering

<http://dbpedia.org/ontology/City> .

Formaliser : The formaliser must convert the interpretation as provided by the previous modules into a formal language, such as SPARQL or SeRQL.

Extending the design, we begin with some default implementations of each module, constituting a limited but functional QALD system we call KQuery.

Default Parser : We use the Stanford Parser (Klein and Manning, 2003) in our default parser module. This provides parts-of-speech as per the Penn Treebank (Marcus et al., 1993) and dependency relations as from **?**. At present it is configured to return only the most likely parse as it deems them but has the potential to return more. It is trained on the English left 3 word dataset for part-of-speech tagging and the English PCFG dataset for dependency resolution. These are both mutable and the parser can be extended to include named-entity recognition.

Default Mapper : Our default mapper explores the dataset for entities with labels including each token text, its lemma, and its synonyms as provided by WordNet (Miller, 1995). A datafile may be provided listing tokens that should not be mapped; typically functional and grammatical words. This mapper may be configured to read instead from a locally-stored "thesaurus", containing pre-defined mappings from tokens to entities, concepts and relations in the data source. These thesauri may also include multiple, weighted mappings for each lexical item.

Default Analyser : The default analyser attempts to extract two pieces of meta-information about the question: the query type and the desired type (also known as an answer type). The query type is a classification of the nature of the question, such as "list", "boolean", etc. The desired type is the class of entity being sought as return values for the question, such as `dbp:City` or `xsd:datetime`.

Default Router : The default router explores the dataset for the smallest paths between mapped entities. It may be configured to cease its exploration after a certain number of iterations (corresponding to the number of connecting entities) in order to prevent excessive response times.

Default Formaliser : We default to formalising the interpretations into SPARQL.

Finally, at the time of writing, one module has an alternative implementation completed. The Default Analyser may be substituted with a dedicated Answer Type Identifier module, as per *citation - under review*. This module requires a pre-trained model consisting of a list of graphs with lemma-POS pairs for nodes and dependency relations for edges. It iteratively checks the question's dependency graph for sub-graph isomorphism with each learned graph and takes the labelled node of the first that matches as the ATI.

4 Conclusion

This deliverable has presented a scenario-based functional specification of the Question Answering and Goal-driven Dataset Summarisation systems that will be developed for the K-Drive project.

Acknowledgement

This research has been funded by the European Commission within the 7th Framework Programme/-Marie Curie Industry-Academia Partnerships and Pathways schema/PEOPLE Work Programme 2011 project K-Drive number 286348 (cf. `http://www.kdrive-project.eu`).

References

Alexopoulos, P., Gomez-Perez, J.-M., and Walker, A. (2012). K-Drive D1.1: Semantic Data. K-drive deliverable, iSOCO, Spain and Aberdeen University, UK.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.