Abstract—Nowadays the internet is becoming a huge dump of documents, links and all other sorts of information. Most common possibilities to explore this information are information retrieval applications such as web search engines. Despite the fact that search engines are doing an excellent job, they still return too much inaccurate information. The solution to this problem can be found in the form of question answering systems, where the user gives a question in natural language, similarly to talking with another person. The answer is the exact information instead of a list of possible results. This paper presents the design of our ontology-driven question answering system with semantic web services support.

Keywords—Ontology, Question answering system, Semantic web, Web services.

I. INTRODUCTION

In today’s world the majority of information is accessible via the World Wide Web. A common way to access this information is through information retrieval applications like web search engines. We already know that web search engines flood their users with enormous amount of data from which they cannot figure out the essential and most important information.

These disadvantages can be reduced with question answering systems. The basic idea of question answering systems is to be able to provide answers to specific question written in natural language. The main goal of question answering systems is to find a specific answer. The answers can be retrieved from domain-specific knowledge corpuses or other external resources like web services.

This article is segmented into eight chapters. The following chapter describes ontologies and semantic description of domain specific knowledge. The third chapter describes the process used for ontology mapping to the relational database.

The following chapter interprets the use of question templates. The fifth chapter describes integration of external knowledge resources. This chapter also explains semantic description of web services. The sixth chapter reveals the importance of user behavior. The seventh chapter describes the architecture and processes of our question answering system. Chapter eight concludes with the summary and suggestions for our future work.

II. SEMANTIC DESCRIPTION OF DOMAIN SPECIFIC KNOWLEDGE

The majority of information available on the web is suitable for human use. This is the main reason why computer applications have a problem understanding of this data [2].

Fortunately, this problem can be solved by using the semantic web. Semantic web is an extension of the World Wide Web. As the name itself suggests, the purpose of the semantic web is to precisely define unambiguous computer understandable metadata, thus enabling computers and people to work in cooperation [4]. One of the most important components of the semantic web are ontologies which can significantly enhance understanding and description of information.

Ontology is a formal representation of knowledge as a set of concepts within a specific domain, and the relationships between those concepts. It is used to reason about the entities within that domain, and may be used to describe the domain. Contemporary ontologies share many structural similarities, regardless of the language in which they are expressed. Most ontologies describe individuals (instances), classes (concepts), attributes, and relations [1].

Web Ontology Language (OWL) is a formal knowledge representation language for authoring ontologies. Ontology languages allow users to write explicit, formal conceptualizations of domains models. The main OWL requirements are: a well-defined syntax, well-defined semantics, efficient reasoning support, sufficient expressive power and convenience of expression. OWL is built on top of RDF and RDF Schema, and uses RDF’s XML syntax. OWL was designed to be interpreted by computers and not for being read by people.

RDF Query Language (SPARQL) is a SQL-like language for querying RDF data.
III. ONTOLOGY MAPPING TO THE RELATIONAL DATABASE

As we describe in the previous chapter, we needed a way to formalize our ontologies. At a beginning we chose OWL for our knowledge presentation. We have been using Protégé, a free, open source ontology editor and knowledge-base framework [3]. It is a great tool for creating semantic web content, but we were concerned with its suitability for our end users. Our users are ordinary people, who do not know anything about ontologies and semantic web. OWL contains much more than we needed for our system. We also needed support for phrases and their synonyms.

All this led us to the conclusion that we have to build our own ontology representation. We took the idea of OWL, which has been reduced with irrelevant elements. We added Domain, Process, Phrases and Synonyms to our solution. We also added the semantic description of methods and parameters, which will be described in detail in the fifth chapter. Our ontology mapping to relational database is shown in the fig.1.

It is important that the system is able to provide answers as fast as is possible. This is the main reason, why we build the whole idea as a relational database solution. We believe that relational database is an optimal solution for us in terms of speed of data searching.

IV. USING QUESTION TEMPLATES

Natural language processing is a domain of Computer Linguistics. Programs and algorithms should behave like they understand natural language [5]. Natural language is ambiguous and contains many synonyms, which can be understood differently, depending on the context of the sentence or even paragraph. The key to understand the importance of the sentence is identification of entities. Methods for determining of the meanings of phrases are generally based on the use of a large knowledge corpus. Most of those methods are slow, since they use a large amount of data and the results are average. This applies to the Slovenian language since there is currently no good enough semantic dictionary for it. Therefore, we used a completely different approach and introduced the question templates in the context of domain-specific knowledge.

Question templates are a bridge between sentences and ontology. They are used as a mapping between relations and objects. Templates can be equated with the ontology as formal presentation skills in the context of a domain. Elements of the question templates are entities composed of phrases, synonyms, class properties or even method parameters. We have basic and complex templates. Basic templates are composed of a question template that is related to a single answer template. In complex cases, the question templates can be related to the template of the second question (sub-questions) if the user didn’t provide enough information for a unique response. All this leads us to question answering dialog.

Example of basic question template:

*What is the e-mail address of [Person_Name] [Person_Surname]?*

*What | is | the | of – words  e-mail address – phrase  [Person_Name] – ontology driven data that represent class Person and its property Name  [Person_Surname] – ontology driven data that represent class Person and its property Surname*

Question template above has only one answer template:

* [Person_Name] [Person_Surname] e-mail address is [Person_Email].

*e-mail address – phrase  [Person_Name] – ontology driven data that represent class Person and its property Name  [Person_Surname] – ontology driven data that represent class Person and its property Surname  [Person_Email] – ontology driven data that represent class Person and its property Email*
V. EXTERNAL KNOWLEDGE RESOURCES

As our system was developed as an applicative project, we had a special request. All information that our question answering system can handle, cannot be presented as an ontology. Certain information must nevertheless be calculated and that means we have to obtain that certain data from external source.

The most logical approach was to use web services. Web services are typically application programming interfaces or Web APIs that are accessed via Hypertext Transfer Protocol (HTTP) and executed on a remote system hosting the requested services.

Because we couldn’t get all the information as web services, we had to extend that process to other ways of calling methods. At the end we added the ability to call local DLL libraries and stored procedures from the database. As all three ways require method calls with the parameters, we need to provide a way to describe the methods and parameters. We consider it ideally to describe them with semantics. So we expanded our ontology representation with method and parameter description.

But of course we also had to develop a special software wrapper that could figure out from the semantic description how to call methods with certain values for parameters and how to return and transform the returned calculated values.

VI. USER TRACKING

In applications that run as web applications we don’t have any control about user inputs. So it is wise to track all user activities during the use of application.

Our system offers active and passive user tracking. Active tracking allows users to tell their opinions about certain answer with simple button click. The system even allows users to comment replied answers. Passive tracking has a whole different approach which is very unobtrusive and users don’t even know about it. Passive tracking uses client-side cookies for anonymous user tracking. We are trying to detect a context switch, which tells us about user’s satisfaction.

With such actions we can constantly update our knowledge database. We can even track user questions and answers returned by our system. If we encounter a certain amount of errors in the responses, we can take appropriate action such as template rebuilding or restructuring of semantic data representation.

On the other hand we can generate all kind of statistics that helps us understand our users’ behavior.

VII. QUESTION ANSWERING PROCESS

Question answering process always starts with the user’s input. The whole question answering process is shown in Fig. 2. When the user enters a sentence, it fires the process for detecting entity candidates. We already know that templates are composed of entities and that’s why we have to find the appropriate candidates for template matching process. Entity candidate detection process uses our domain specific knowledge database for detecting entities. Entities are recorded as instances of our ontology. At that stage our process uses a dialog states table, where the actual state of user dialog is stored. That is very important for understanding what data has already been entered by the user.

The most important task in the whole process is the template matching process. This process must decide which question template is the most similar to the sentence entered by the user. Entity candidate and dialog state list help that process to find the best calculated question template. Question templates are also ranked, which helps us to restrict our choice. In case the system couldn’t find the exact template match, it is capable to advice the user which question template to use.

Now, when the appropriate question template is found, we can generate an answer related either to an answer or a sub-question template.

If entity in the template is represented as an external information resource, we have to find semantic description of that source in our ontology. External information resources could be a web service, a DLL library or a stored procedure in a database. A specifically written software wrapper then calls an appropriate method, which returns the result that represents the required entity values.

Before the answer is shown to the user, a special process records the user activity and alters the dialog state table. If dialog with the user is not finished yet, an answer is formulated as a question. At that stage we have entered in the question answer dialog.

VIII. CONCLUSION

This article describes our ontology-driven question answering system with semantic web services support. While
we didn’t want to build large knowledge corpuses of Slovenian language, we decided to semantically describe our domain-specific knowledge. The key component to our system is a well defined and semantically described ontology based knowledge database. Although there are some methods for storing ontologies, we built our own ontology mapping to relational database.

Because the question answering system should somehow understand natural language we managed to provide question templates. Question templates are a bridge between sentences and Ontology. The template matching process is the most important part in our system. This process is responsible for the entire conversation dialogue. The answer generation process is also built on top of the question templates. Some entities in those templates should be filled from ontology instances or even from external knowledge resources like web services. Our question answering system also tracks user’s behavior.

A challenge for our future work is to improve algorithms for entity candidate detection and to speed up the algorithm for finding the minimum distance in question templates. A special treatment will be given to expand the set of external resources.

Our ontology based knowledge database should always grow. You can get good results only if you have a large enough and quality knowledge corpus.

REFERENCES