

Development of Q&A Systems Using AcQA

Renato Preigschadt de Azevedo¹ 

Centro Algoritmi (CAIlg-CTC), Department of Informatics, University of Minho, Braga, Portugal
<https://acqa.di.uminho.pt>
renato@redes.ufsm.br

Maria João Varanda Pereira 

Research Centre in Digitalization and Intelligent Robotics (CeDRI),
Instituto Politécnico de Bragança, Portugal
mjoao@ipb.pt

Pedro Rangel Henriques 

Centro Algoritmi (CAIlg-CTC), Department of Informatics, University of Minho, Braga, Portugal
pedrorangelhenriques@gmail.com

Abstract

In order to help the user to search for relevant information, Question and Answering (Q&A) Systems provide the possibility to formulate the question freely in a natural language, retrieving the most appropriate and concise answers. These systems interpret the user question to understand his information needs and return him the more adequate replies in a semantic sense; they do not perform a statistical word search like happens in the existing search engines. There are several approaches to developing and deploying Q&A Systems, making it hard to choose the best way to build the system. To turn easier this process, we are proposing a way to automatically create Q&A Systems (AcQA) based on DSLs, thus allowing the setup and the validation of the Q&A System independent of the implementation techniques. With our proposal (AcQA language), we want the developers to focus on the data and contents, instead of implementation details. We conducted an experiment to assess the feasibility of using AcQA. The study was carried out with people from the computer science field and shows that our language simplifies the development of a Q&A System.

2012 ACM Subject Classification Software and its engineering → Source code generation; Software and its engineering → Domain specific languages; Software and its engineering → Design languages

Keywords and phrases Question & Answering, DSL, Natural Language Processing

Digital Object Identifier 10.4230/OASICS.SLATE.2020.8

Funding This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the Projects Scopes: UIDB/05757/2020 and UIDB/00319/2020.

1 Introduction

With the increased usage of personal assistants, which allow the user to ask questions and get answers from various subjects, these types of systems are being used by a large number of people. Approximately forty or fifty years ago, studies began on Q&A (Question and Answering) Systems [39, 9, 33, 16], but due to computational limitations, these systems had limited scope. Some Q&A Systems were more or less successful; some of them were discontinued, demonstrating the difficulty of building and maintaining a system capable of understanding natural language as a human can. Even so, demand increased at a high pace, leading to researches to build better and more efficient systems.

¹ corresponding author



Questions are asked and answered several times per day by a human. Q&A Systems try to do the same level of interaction between computers and humans. This approach differs from standard search engines (Google, Bing, and other search engines) because it makes an effort to understand what the question expresses and try to give concise answers instead of using keywords from the question asked and provide documents as results.

A simple Q&A System is composed of several processes: question analysis, query processing (extraction of potential answers), and answer formulation [11]. To be able to create a functional Q&A System, all these processes have to be carefully chosen by the domain specialist and implemented by the programmer. The programmer has to be an expert in the chosen programming language, as well as the various libraries required to implement the system (Natural Language Processing, Neural Networks, Database Access, among others). These complex systems increase the possibility of errors occurring, making the implementation process more time consuming and costly.

Domain-specific Languages (DSLs) can simplify and accelerate the development of applications. The development of a DSL to construct a Q&A System allows the user to specify components used in these systems more abstractly. This approach makes the process of implementing the system more straightforward and less error-prone. To the best of our knowledge, this is the first work that uses DSL to create Q&A Systems. We did not identify any similar work to compare and discuss.

Taking into account the importance of Q&A Systems and the complex tasks that must be implemented to create such an intelligent tool, the goal for this paper is to discuss the design of AcQA (Automatic creation of Q&A Systems) DSL to make the development of these systems easier following a systematic and rigorous approach.

The structure of the paper is as follows: Q&A Systems are discussed in Section 2; in Section 3 some relevant aspects on the use of DSLs for automatic construction of language-based tools are presented; AcQA language is then presented in Section 4; Section 5 describes an experiment and discusses the results; and in Section 6 conclusions are presented and directions for future work are proposed.

2 Question & Answering Systems

The wideness of information available associated with the demand for direct answers from the users requires a different approach from standard search engines. The use of natural language to communicate with computer systems turns the information technology useful for all types of persons, allowing them to specify deeply the information needed. Q&A Systems are not new, but they still need to be improved in terms of answers accuracy, and in terms of knowledge domains. The main idea of these systems is to receive the user question and analyze not only the keywords but also the intention. To discover the intent within a question is not an easy task. Besides the capability of understanding the user question, the system must retrieve the best possible answers, ranking them.

There are several approaches in the literature explaining the construction of Q&A Systems [15, 25, 36, 38], explaining the typical sequence of development stages. At first, several technical approaches should be carefully studied to allow the processing of natural language. In the past, small Knowledge Bases were used, allowing the construction of simple Q&A Systems. These systems used simple schemas with a small number of entities and relations, ad-hoc approaches (manually constructed rules), among other strategies to create the knowledge base (KB). The KB was specifically tailored to the specific domain, needing much effort from the original designers of the Q&A System to add new content or add a new domain. These strategies do not support the construction of scalable systems and turn complex the development of open domain systems.

To construct a Q&A System, we need to develop three processes: question analysis, query processing (extraction of potential answers), and answer formulation. The techniques used for question analysis seeks to recover meaning from the input text, sometimes employing Natural Language Processing (NLP) to achieve the goal. Natural Language Processing is an area of computation that includes parsing, part-of-speech (POS) tagging, statistical models, ontologies, and machine learning [24]. Pattern matching and the use of tags can also be used to process the input text. Query processing approaches are responsible for handling the input text to create the queries necessary to extract relevant information from the KB. The answer formulation uses information gathered in question analysis and query processing to generate or retrieve possible answers.

The result of a Q&A System can be fragments of documents, a list of links to web pages, images, a simple and concise sentence, or a ranking of sentences. This work is focused on systems that retrieve one answer or a rank-ordered list of answer candidates.

Q&A Systems are classified as being closed or open domain. In open domain Q&A Systems, the questions are domain-independent, and the system aims to answer anything that the user asks. In this case, an extensive repository of information must be used, and the system must be able to answer questions of all kinds of subjects. Restricted domain Q&A Systems (closed-domain) work only with a specific domain, not being able to answer questions outside the proposed field. The information repository is based on a well-defined knowledge base, made out of data only related to the proposed field, being able to achieve better accuracy than open domain Q&A System.

Some examples of open domain Q&A Systems are Intelligent Q&A System based on Artificial Neural Network [2], Automatic Question-Answering Based on Wikipedia Data Extraction [22], A Content-Aware Hybrid Architecture for Answering Questions from Open-domain Texts [21], WolframAlpha [23] and IBM Watson [15]. Some examples of Closed domain Q&A Systems are Q&A System on Education Acts [28], Python Question Answer System (PythonQA) [35], and K-Extractor [4].

2.1 Available Q&A Systems

There are several Q&A Systems using NLP to process the user input data and provide answers accordingly. In the work proposed by Ramos et al. [35], the PythonQA system was developed to answers questions about the programming language Python. It was developed in the Python programming language, together with some libraries such as Natural Language ToolKit (NLTK) [8], Django, among others. The system processes the input from the user dividing a phrase into several components and trying to identify three main elements: action, keywords, and the question type. Then, these three elements are compared to the knowledge base, built with data from the Python Frequently Asked Questions, to retrieve and show answers to the users of the Q&A System.

MEANS [6] also used NLP to process the corpora and user questions. The medical subject is the domain that was used by the authors of this Q&A System. The knowledge base was created through sources of RDF annotated documents, based on an ontology. To provide better answers, the authors propose ten question types to classify user questions.

Other works use ontologies as KB, as in [34], [30], [7], [26]. The authors in [34] propose a Q&A System that uses ontology assistance, template assistance, and user modeling techniques to achieve 85% of accuracy in their experiments. The authors of [30] propose an algorithm to automatically update the ontology used by the system and use a semantic analyzer that operates on an ontology to extract answers.

8:4 Development of Q&A Systems Using AcQA

To be able to answer questions about the United Kingdom Parliament education acts, a Q&A System was proposed in [28]. The knowledge base was made from the data publicly available from the United Kingdom parliament using NLP techniques. This proposed Q&A System uses only keywords from the user question, ignoring the question type and other pieces of information present in the user input text.

Some works use artificial intelligence (AI) to process questions and create the knowledge base for the Q&A System. In [10], a Q&A System, called AskHERMES, is proposed to solve complex clinical questions. The authors use five types of resources to construct the knowledge base (MEDLINE, PubMed, eMedicine, Wikipedia, and clinical guidelines). The user input question is classified by twelve general topics, made by a support vector machine (SVM). The authors process possible answers through a question summarization and answer presentation modules based on a clustering technique. Another work using AI is proposed by Weissenborn et al. [40], where the authors propose a Q&A System based on fast neural network techniques. According to results from their proposed system, the system uses a simple proposed heuristic to achieve the same performance compared to more complex systems.

There are Q&A Systems that work with different input, like images. In [14], an approach to generate image descriptions automatically is proposed. Firstly words describing the image are discovered and stored. Secondly, it generates sentences associating the objects in the picture. The final step is to rank the phrases according to the MERT [31] model to present the sentences to the user.

As it was said, the most well-known examples of open domain Q&A Systems are Wolfram[23] and IBM Watson [15]. WolframAlpha and IBM Watson are examples of this type of Q&A Systems. WolframAlpha is a well-established Q&A System to answer questions of any type and was initially a closed domain Q&A System to answer questions about mathematics. It allows the user to extend the version available online with the pre-existing knowledge base or to upload data through a paid subscription. IBM Watson is a Q&A System that was initially created to play in the Jeopardy TV quiz program. Watson is now an Artificial Intelligence framework provided by IBM. It allows working with a variety of areas, such as Q&A Systems and natural language processing. Watson is available through paid subscriptions.

3 Domain Specific Languages (DSL)

Domain-Specific Languages (DSLs) can simplify the development of applications [1]. According to Fowler [17], Domain-Specific Language is a computer programming language of limited expressiveness focused on a particular domain. Although the handicap of having to learn a new language[29], Domain-Specific Languages (DSLs) can accelerate and make simple the development of applications [1].

DSLs are relevant to developers for two main reasons: improving programmer productivity and allowing programmers and non-programmers to read and understand the source code. The improved programmer productivity is achieved because DSLs try to resolve a minor problem than general-purpose programming languages (GPL) [18], making it more straightforward to write and modify programs/specifications.

Since usually DSLs are smaller than GPLs, they allow domain specialists to see the source code and get a more general view of their business. DSLs offer the capacity to domain specialists to create a functional system, with no prior knowledge of GPLs.

What distinguishes DSLs from GPLs is the expressiveness of the language. Instead of providing all the features that a GPL must contain, such as supporting diverse data types, control, and abstraction structures, the DSL has to support only elements that are necessary to a real domain. Examples of commonly used DSLs, according to [18], are SQL, Ant, Rake, Make, CSS, YACC, Bison, ANTLR, RSpec, Cucumber, HTML, LaTeX.

Generative programming is related to the construction of specialized and highly optimized solutions through a combination of modules to decrease the conceptual gap between coding and domain concepts. This approach simplifies the management of several components, increasing efficiency (space and execution time) [13].

In that sense, generative programming recommends to apply separation of concerns, namely, to deal with one important element at a time, and combine these elements to generate a component; There is a clear separation from the problem and solution. Each component can be adapted for different scenarios using parameterization and creating different families of components, implying the management of dependencies and interactions to combine them.

Generative programming uses DSL at a modeling level [12] to allow users to operate directly with the domain concepts, instead of dealing with implementation details of GPLs. The system can automatically generate executable code from a textual or graphical DSL specification [13].

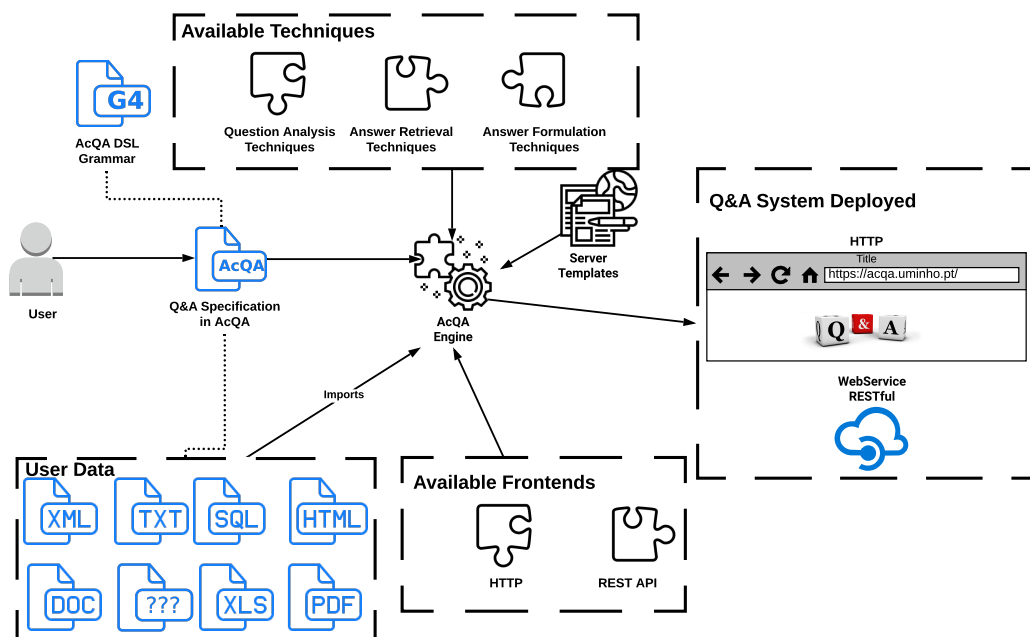
In this work, the concept of generative programming is applied through the use of a DSL to allow the automatic creation of Q&A Systems, avoiding the need to build the system line-by-line. A compilers generator is used to process the DSL formal specification (grammar) to build a Q&A System compiler. That new compiler will automatically produce the desired Q&A System taking into account the input description written using the DSL proposed. Section 4 presents the proposed approach.

4 AcQA – Automatic creation of Q&A Systems

The use of generative programming and domain-specific languages allows domain specialists to develop entire Q&A Systems without the need to code or have knowledge in GPLs. We used these concepts to enforce constraints and validate inputs, allowing domain specialists to build the Q&A System focusing on what techniques to use, rather than how to implement them.

In this Section, the domain-specific language AcQA (Automatic creation of Q&A Systems) is presented. AcQA allows an expert or regular user to specify this kind of system more straightforwardly than in a GPL. The focus is on the behavior of the whole system, rather than how to implement them. This approach allows the user to focus on the data that are made available for building the system knowledge base as well as change behaviors such as the techniques that need to be used to process the user inputs (questions) and its front-end (Web, Webservice, others). Experienced Q&A Systems developers who already have tools to construct Q&A Systems can extend the AcQA language to use these tools. Domain experts who have no previous experience developing Q&A Systems can develop a functional Q&A System using only the AcQA DSL.

To achieve the objective of generating a Q&A System, we propose to apply the concepts discussed in the Sections 2 and 3. We developed the grammar for AcQA DSL to be recognized by the AcQA Engine. The AcQA Engine uses the ANTLR [32] tool to process the user specification written in AcQA DSL. This engine is responsible for recognizing specifications written in AcQA and preparing all the required configuration and code generation needed to run the Q&A System. Our work generates Python code to handle the tasks needed both by the compiler and the generated Q&A System.



■ **Figure 1** AcQA architecture.

Figure 1 depicts the steps needed to generate the Q&A System and describe how they are connected. First, the user writes a specification of the desired Q&A System in AcQA. It has to define at least three essential elements: the server where the system is going to be deployed, the input data to generate the knowledge base, and which user interface to use. These items and options are explained and discussed in Section 4.1. After the specification written by the user is validated, the AcQA Engine starts to generate the code for the Q&A System. First, the Engine generates the techniques needed to process the inputs from the user of the Q&A System. Secondly, the code for the front-end (user interface) is generated. The third step is to deploy the generated Q&A System into the server. When the server is ready, the data required to make the KB is supplied to the Q&A System to produce the initial KB. After these steps, the Q&A System is ready to process questions and provide answers to the final users. The KB is build by the natural language processing algorithms defined in the AcQA specification. More details on the techniques used to generate the KB are provided in the paper [3].

It is also possible for a user with proper knowledge of GPL and natural language processing techniques to implement new algorithms extending the existing techniques through an interface available in AcQA. The user can choose which parser is used for the system access and understand the data. The data is then processed by the AcQA Engine to generate the knowledge base used by the Q&A System. This module resorts to general-purpose parsers available to import XML, TXT, SQL, or customized parsers to deal with proprietary data formats.

The AcQA Engine processes all specified options to create a Q&A System accordingly to available front-ends (HTTP and Representational State Transfer (RESTful) web service). With the help of Server Templates, the AcQA Engine connects to a server and installs and configures all the requirements needed to deploy the Q&A System.

```

1      acquFile: lines+ EOF;
2      lines:(comment | decl | TERMINATOR | EOF) ;
3      decl: inputfile | techniques | ui | server | nodeploy |
         cleankb;
4      inputfile: INPUT '(' PATH (',' input_options)* ')';
5      techniques: TECHNIQUES '(' TECHNIQUES_TYPE (','
         techniques_options)* ')';
6      ui: UI '(' UI_TYPE (',' ui_options)* ')';
7      server: SERVER '(' HOST (',' server_options)* ')';
8      techniques_options: params;
9      server_options: (USER | PASSWORD | KEY )'= ' value;
10     ui_options: params;
11     input_options: INPUT_PARSER | params;
12     params:key '= ' value;
13     key: IDENTIFIER;
14     value:NUMERIC_LITERAL | STRING_LITERAL | INT | PATH;
15     comment:COMMENT | SINGLE_LINE_COMMENT | MULTILINE_COMMENT;
16     PATH: STRING_LITERAL;
17     STRING_LITERAL: '\'' ( ~'\'' | '\'\'' )* '\'';
18     SINGLE_LINE_COMMENT: '--' ~[\n]*;
19     MULTILINE_COMMENT: '/*' .*? ( '*/' | EOF );

```

■ **Figure 2** AcQA grammar fragment.

To conclude, AcQA language is an external DSL, containing a custom syntax to make the specification and parameterization of a Q&A System more friendly for the user. There is also a default value for parameters, thus allowing the user to specify only a few values and build the Q&A System automatically. In Section 4.1, we present the main sections of AcQA grammar, and in Section 4.2, we present an example of a Q&A System written in AcQA syntax; the idea is to give more details on the AcQA language and illustrate its use.

4.1 DSL Design

The AcQA language already has several off-the-shelf elements to allow the construction of a Q&A System. This Section presents the elements available to be used by the Q&A System creator.

Figure 2 shows the main elements of AcQA grammar. The listing in that figure shows the declaration of the main definitions needed to set up the initial working Q&A System. Each line of a specification in AcQA can have a comment or a declaration (lines 1 and 2). AcQA DSL has six declaration blocks (line 3): Input File, Techniques, UI, Server, NoDeploy, and CleanKB. These blocks specify the behavior of the generated Q&A System.

The input file (line 4 in Figure 2) specifies the data that has to be imported to create the knowledge base of the Q&A System. Line 4 shows that AcQA specification needs the user to set the path where the input file is located. The input block also allows the user to set optional parameters (line 11) to change the parser's behavior, such as parser type, parser options. There are several parser types for parsing the user data needed to build the knowledge base. In this first release of AcQA, it is possible to parse the following file formats: eXtensible Markup Language (XML), Raw Text (in any encoding, as long as it works with Python), SQL, HTML, DOC, XLS or PDF. Other file formats can be processed through the extension of an interface provided by the AcQA language, and it is the developer's responsibility to program this extension. The optional parameters are in the form of *key => value* (as defined in lines 12-14) to change the behavior of the parser.

The Techniques block (line 6 in Figure 2) defines which techniques are used in the question analysis, answer retrieval, and answer formulation processes. If this block is not specified, the default behavior is to use techniques associated with the Triplets approach. These Triplets techniques are initially based on works described in [3] and [35], where a closed-domain Q&A System was developed. These techniques were used as the initial approaches to accelerate the development of AcQA DSL and use the know-how from our language processing group (gEPL). The not obligatory options are defined as *key => value*.

The UI block is responsible for specifying which type of UI the system deploys (line 6 in Figure 2). The initially available front-ends to provide access to the Q&A System are twofold: HTTP and RESTful Webservice. The HTTP front-end is a graphical interface available through the HTTP protocol, having a responsive interface and can be accessed through computers, tablets, or cell phones. Using the RESTful front-end allows the creators of the Q&A System who already has some developed platform to provide access to the user who wants to ask questions, by integrating their platform with the generated Q&A System. Figure 5 shows the HTTP front-end visualization in a computer and on a smartphone. The block of AcQA that configures and deploys the system to a given location is the Server (line 7). The user needs to specify the server's hostname, the user name, and password, or the RSA key to access the server.

The last two declaration types are reserved words and change the way that the system is deployed, as shown in Figure 2 (end of line 3). If the user does not specify the keywords NoDeploy or CleanKB, the AcQA Engine deploys all the generated code and sends the input data to be processed by the Q&A System. All previous configurations are lost if there is already a running instance of the Q&A System in the server. The keyword NoDeploy does not reinstall the Q&A System and maintain all data already present on the server. When the developer wants to empty the KB, they can use the keyword CleanKB.

All the parameters written in AcQA are syntactically and semantically validated. For example, the parameters path and hostname are syntactically verified in the grammar of AcQA. AcQA Engine is responsible for enforcing semantic correctness.

4.2 Specifying a Q&A System for Board&Card Games in AcQA DSL

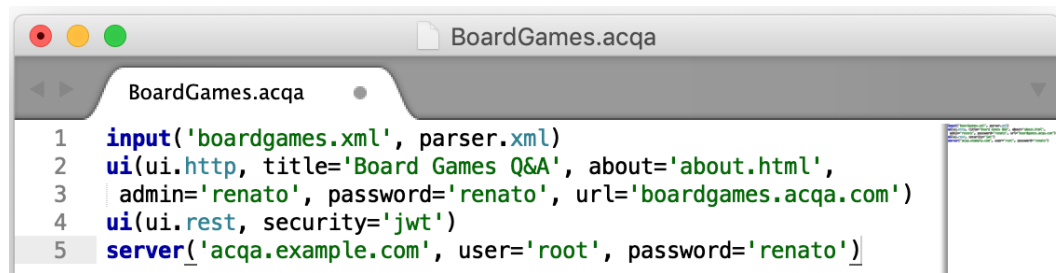
This Section illustrates AcQA DSL syntax by specifying a board & card games Q&A System. The language syntax of AcQA was defined to be simple, allowing the user to create a specification of a Q&A System without the need to have prior knowledge on GPL's. The language syntax also allows the user to parameterize the techniques used to build the knowledge base, process answers, and other parameters.

The Q&A System specified in AcQA is intended to answer questions concerning board & card games. This subsection gives an overview of the domain of the Q&A System. It also discusses the data used in the running example.

According to [19], board games are games with a fixed set of rules that limit the number of pieces on a board, the number of positions for these pieces, and the number of possible moves. There are several discussion groups on the Internet about these types of games; they allow users to ask questions about the rules, strategies, or even questions about the games. An example of one question about the game *Exploding Kittens* is: *How many persons can play the original game?*. In this running example, we use data available from StackExchange (SE)² in Archive.org³.

² www.stackexchange.com

³ <https://archive.org/details/stackexchange>



```

1 input('boardgames.xml', parser.xml)
2 ui(ui.http, title='Board Games Q&A', about='about.html',
3   admin='renato', password='renato', url='boardgames.acqa.com')
4 ui(ui.rest, security='jwt')
5 server('acqa.example.com', user='root', password='renato')

```

■ **Figure 3** Board and Card Games Q&A System specification in AcQA.

StackExchange (SE) is an Online Social Question and Answering site which allows users to post questions and answers, in this case, handwritten into the system by other community members. Card & Board Games is one of the 166 Stack Exchange Community. We decided to analyze the Card & Board Games among another Community Question Answer site (CQAS) because of the public availability of the data, as well as being regularly updated.

The data used in this work were made available on 2019-03-04 and had approximately 40,7 MB of size and 31395 Posts (questions or answers). The data from SE was preprocessed to extract Questions that have answers from the data dump file.

Figure 3 presents a code fragment of a specification written in AcQA DSL to configure a Q&A System for Board Games. Line 1 in Figure 3 specifies the file name (with a valid path) and which parser is used to process and load the user data into the knowledge base of the Q&A System. It is possible to set optional variables to determine a specific behavior of the parser. This input file is processed by the AcQA Engine and is sent to the deployed Q&A System through the RESTful web service (specified in line 4). After receiving this file, the Q&A System start a task to process the records and generate the knowledge base. In this example, the techniques block was omitted, so the generated Q&A System uses the triplets technique by default.

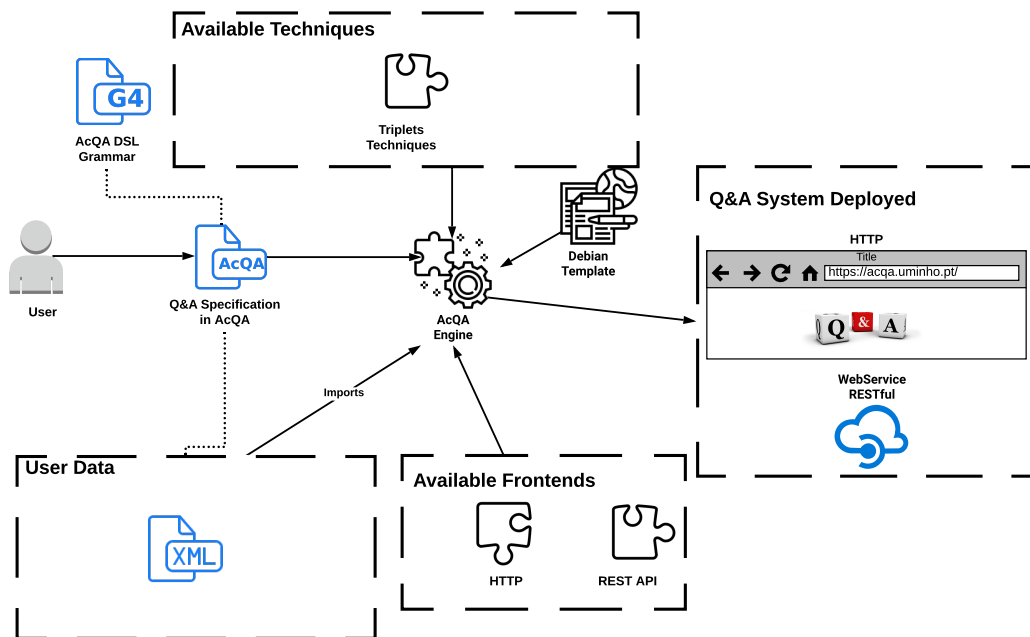
The UI's are specified in lines 2–4. It is obligatory to specify at least one parameter: the UI type. In lines 2-3, an HTTP UI is defined with some parameters of the UI: the title of Q&A System, the HTML used in Section About, the admin credentials to access the admin page, and the URL used to configure the services in the Server, respectively. In line 4, a setup of a RESTful web service is shown. The parameter security defines which type of security is used by the RESTful web service. In this example, the JSON Web Token[5] is used to provide authentication and authorization in the web service.

The code at line 5 (Figure 3) configures in which Server the Q&A System is deployed, and the first parameter is the Server hostname. The last two parameters are the login information that is used to connect to the server through an SSH (Secure Shell) protocol [41]. The password parameter can be interchanged by the RSA (Rivest-Shamir-Adleman) cryptographic key for added security [37]. There is also an optional parameter to specify which type of the Server (Debian, Fedora, among others). Currently, the default value for the server is Debian-like⁴.

The AcQA DSL allows the user to change the behavior of the whole Q&A System. For example, the user can change the language of the system to Portuguese instead of English using the parameter *language="Portuguese"* inside the UI block. The changes are applied in tokenizer, POS (Part-of-Speech) tagger, lemmatizer, and Wordnet language.

⁴ <https://debian.org>

8:10 Development of Q&A Systems Using AcQA



■ **Figure 4** Steps to create a Q&A System.

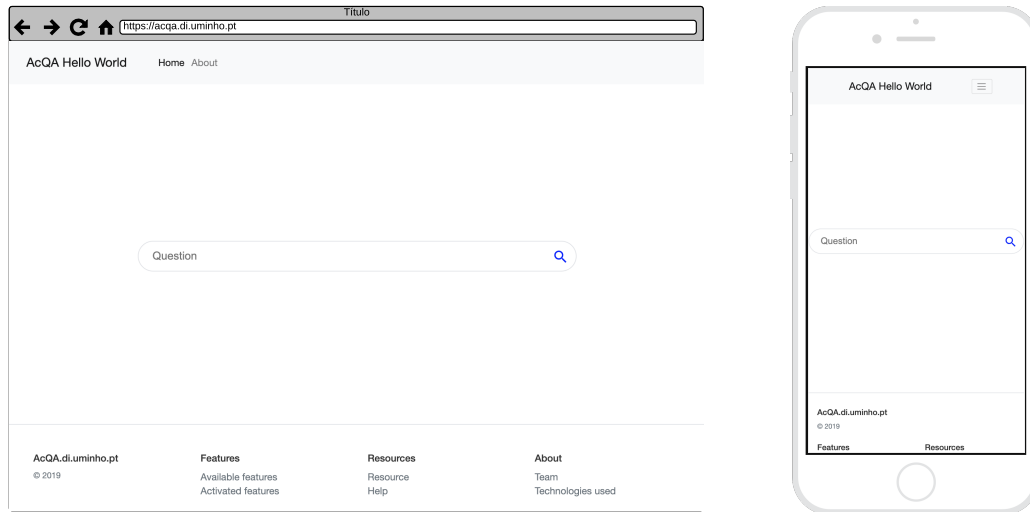
The steps needed to generate the fully operational Q&A System are depicted in Figure 4. The AcQA grammar is processed by the compiler from AcQA to generate the AcQA Engine. The specification is written by the user and recognized by the AcQA compiler. The data from the user is imported through the Input Parser set in the AcQA specification. The AcQA Engine then generates a Q&A System specified by the user. The Q&A System is deployed into the Server when AcQA Engine executes the Deploy Engine. After that, the Deploy Engine process the User data to create the knowledge base that is used by the fully operational Q&A System.

When a user of the Q&A System accesses the provided URL in the AcQA specification (in our example: boardgames.acqa.com), an HTTP Web UI is displayed, as shown in Figure 5. If more than one answer is generated, the answers are presented according to the ranking generated by the trust value.

5 Assessment

In order to assess the use of AcQA language and test the performance and usability of the generator implemented, an experiment was designed and conducted. This experiment aims at recognizing if it is feasible to use a language like AcQA to create in short time and with a minimum effort Q&A Systems.

The participants in the experiment received a description of a possible scenario within which they were supposed to create, resorting to AcQA, a Q&A System to answer questions about a specific domain.



■ **Figure 5** Screenshot of the Board Games Q&A System generated by AcQA.

5.1 Participants

This experiment was applied to people with distinct education, reaching undergraduate, M.S., or Ph.D. students, masters, and doctors. All the participants (actually 17) are from the computer science area and have prior programming experience, ranging from beginners to experts. There were seventeen participants that successfully answered the survey in this experiment.

5.2 Hypotheses definition

The experiment was planned to understand if the following research questions (RQ) are true:

- **RQ1.** Does the AcQA usage help to understand Q&A Systems design?
- **RQ2.** Does the AcQA usage affect the time required to deploy a Q&A System?
- **RQ3.** The tools provided with AcQA can successfully assist the user in the development of the Q&A System?
- **RQ4.** Can AcQA effectively help the user in the deployment of the Q&A System?

5.3 Experiment Design

AcQA language was introduced to the participants through a tutorial on the development and deployment of a complete Q&A System. This tutorial provides an example of a specification written in AcQA to produce a Q&A System to work in the *board & card games* domain. It explains the input file needed to create the knowledge base of the Q&A System. As input file, the participant can choose among seven XML files, extracted from Stack Exchange data dump (as well as the example presented in Section 4.2): cooking, DIY, fitness, hardware, lifehacks, mechanics, parenting. Credentials to get into a Windows Remote Desktop Connection were provided to each participant. This remote connection gives access to a fully functional Integrated Development Environment (IDE) based on Sublime Text Editor, configured to provide support and syntax highlighting for development under AcQA eco-system.

Credentials to a clean install of a Linux server (Ubuntu Server) are also provided to each participant; under that server, Q&A System can be deployed and tested.

8:12 Development of Q&A Systems Using AcQA

After writing his AcQA specification and after building and testing the Q&A System generated by AcQA Engine, the participant was requested to answer a survey organized in four parts: section one contains a questionnaire that gathers information about the participants' prior experience and academic background; section two collects information about the participant's programming experience; section three asks the subject about his experience in the design and development of Q&A Systems; and finally, Section four enquires the participant about AcQA usage.

The experiment here described is accessible at <https://acqa.di.uminho.pt/experiment/>. It presents the tasks needed to evaluate AcQA language and is available to anyone that wants to participate and send us feedback about the experience, thus helping the development of the language.

5.4 Experiment results

As said above, a first instance (others are under preparation) of the described experiments was conducted recently involving seventeen experienced computer science participants, one undergraduate student, four Ph.D. students, three masters students, six masters, and three Ph.D. According to the answers, the majority of participants self-declared as beginners concerning the development of Q&A Systems.

The answers provided by the participants in this experiment were subjected to a reliability test using the Cronbach alpha scale. As the values for the Cronbach alpha scale computed were higher than the threshold of 0.66, the reliability of the measuring instrument can be confirmed.

Some participants stated that the support tools provided (syntax highlighting and build help on a code editor) made it easier to program and understand the AcQA code.

Table 1 shows the average, median, and standard deviation of the respondent answers, quantified in a 1–5 scale.

The research question 1 (Does the AcQA usage help to understand Q&A Systems design) got an average rate of 4, mostly because the respondents did not have prior experience (52.94% of respondents) or are beginners (35.29%) developing Q&A Systems. Only 5.88% of participants stated that were regular Q&A System developers, and 5.88% self-declared as experts.

Analysing the experiment outcomes and the answers collected from the seventeen questionnaires, it is fair to conclude that the four research questions were positively confirmed: (1) the exercise of writing a specification in AcQA DSL contributes for a clear understanding about the design of a Q&A System; (2) using AcQA specification language and engine the time required to deploy a Q&A System is reduced; (3) the editing tools provided with AcQA actually aid the user during the development of a Q&A System; (4) resorting to AcQA system, the deployment of a Q&A System becomes easier and faster.

These statements need further validation, but we are recruiting more people to participate in the next experiments. Anyway, at present, it is obvious that the results so far attained are promising.

⁵ A five-grade scale, starting from strongly disagree (1) to strongly agree (5) was used in the questionnaires

■ **Table 1** Statistics about the answers to the research questions (N=17).

	Average ⁵	Median	St. Dev.
RQ1	4.00	4.00	0.50
RQ2	3.66	3.66	0.50
RQ3	3.91	4.00	0.41
RQ4	4.23	4.50	0.49

6 Conclusion

In this paper, we presented the domain-specific language AcQA, which allows for the specification of a Q&A System. The specification in AcQA does not require that the developer has a deep knowledge of general programming languages; instead, the developer can focus on the choice of most convenient techniques to include in his new Q&A System.

To show the viability of the proposed approach and expressiveness of AcQA language, a Q&A System to answer questions about *Board Games* was developed and described in the paper. The example used data extracted from StackExchange to create the knowledge repository; its implementation showed that it is possible to generate a Q&A System without the need for complicated and costly GPL development.

From the analysis of the experiment outcomes, we can conclude that the majority of the respondents agree with all the research questions. Moreover, we observed that every participant was able to successfully create and use a Q&A System during the experimental session. We created a website at <https://acqa.di.uminho.pt/> to describe the project and provide documentation and tools that support the AcQA language.

As previously said, in the near future we will conduct more experiments with AcQA eco-system, involving final users from different areas and with different backgrounds (non-programmers, domain specialists, etc.). These experiments will be tuned to test the usability and usefulness of the AcQA DSL.

As future work, we want to extend AcQA language to integrate more techniques [6, 40, 20, 27] for query interpretation and answer formulation, to improve the user interface, and to include server templates. Another direction for future research regards the improvement of the techniques' customization, allowing the developer to make more significant changes in the algorithms and parameters while he writes an AcQA specification.

References

- 1 Sorin Adam and Ulrik Pagh Schultz. Towards tool support for spreadsheet-based domain-specific languages. In *ACM SIGPLAN Notices*, volume 51, pages 95–98. ACM, 2015.
- 2 Ahlam Ansari, Moonish Maknojia, and Altamash Shaikh. Intelligent question answering system based on Artificial Neural Network. In *2016 IEEE International Conference on Engineering and Technology (ICETECH)*, pages 758–763. IEEE, March 2016. doi:10.1109/ICETECH.2016.7569350.
- 3 Renato Azevedo, Pedro Rangel Henriques, and Maria João Varanda Pereira. Extending PythonQA with Knowledge from StackOverflow. In Álvaro Rocha, Hojjat Adeli, Luís Paulo Reis, and Sandra Costanzo, editors, *Trends and Advances in Information Systems and Technologies, WorldCist2018*, volume 745 of *Advances in Intelligent Systems and Computing*, pages 568–575. Springer International Publishing, 1 edition, 2018. doi:10.1007/978-3-319-77703-0_56.
- 4 Mithun Balakrishna, Steven Werner, Marta Tatu, Tatiana Erekhinskaya, and Dan Moldovan. K-Extractor: Automatic Knowledge Extraction for Hybrid Question Answering. In *Proceedings – 2016 IEEE 10th International Conference on Semantic Computing, ICSC 2016*, 2016. doi:10.1109/ICSC.2016.30.

- 5 Victoria Beltran. Characterization of web single sign-on protocols. *IEEE Communications Magazine*, 54(7):24–30, 2016.
- 6 Asma Ben Abacha and Pierre Zweigenbaum. MEANS: A medical question-answering system combining NLP techniques and semantic Web technologies. *Information Processing and Management*, 51(5):570–594, 2015. doi:10.1016/j.ipm.2015.04.006.
- 7 Ghada Besbes, Hajer Baazaoui-Zghal, and Henda Ben Ghezela. An ontology-driven visual question-answering framework. *Proceedings of the International Conference on Information Visualisation*, 2015-Sept:127–132, 2015. doi:10.1109/iv.2015.32.
- 8 Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O’Reilly Media, Inc., 1st edition, 2009.
- 9 Daniel G Bobrow. A question-answering system for high school algebra word problems. In *Proceedings of the October 27-29, 1964, fall joint computer conference, part I*, pages 591–614. ACM, 1964.
- 10 Yong Gang Cao, Feifan Liu, Pippa Simpson, Lamont Antieau, Andrew Bennett, James J. Cimino, John Ely, and Hong Yu. AskHERMES: An online question answering system for complex clinical questions. *Journal of Biomedical Informatics*, 44(2):277–288, 2011. doi:10.1016/j.jbi.2011.01.004.
- 11 Alexander Clark, Chris Fox, and Shalom Lappin. *The Handbook of Computational Linguistics and Natural Language Processing*. Wiley-Blackwell, 2010.
- 12 Pierre Cointe. Towards generative programming. In *Unconventional Programming Paradigms*, pages 315—325. Springer, 2005.
- 13 Krzysztof Czarnecki. Overview of generative software development. In *Unconventional Programming Paradigms*, pages 326–341. Springer, 2005.
- 14 Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K. Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. From captions to visual concepts and back. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:1473–1482, 2015. doi:10.1109/CVPR.2015.7298754.
- 15 D Ferrucci. Build watson: An overview of DeepQA for the Jeopardy! Challenge. In *2010 19th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, page 1, 2010.
- 16 Lance Fortnow and Steve Homer. A short history of computational complexity. Technical report, Boston University Computer Science Department, 2003.
- 17 Martin Fowler. *Domain-specific languages*. Pearson Education, 2010.
- 18 Debasish Ghosh. *DSLs in action*. Manning Publications Co., 2010.
- 19 Fernand Gobet, Jean Retschitzki, and Alex de Voogt. *Moves in mind: The psychology of board games*. Psychology Press, 2004.
- 20 David C Gondek, Adam Lally, Aditya Kalyanpur, J William Murdock, Pablo Ariel Duboué, Lei Zhang, Yue Pan, Zhao Ming Qiu, and Chris Welty. A framework for merging and ranking of answers in deepqa. *IBM Journal of Research and Development*, 56(3.4):14–1, 2012.
- 21 Md Moinul Hoque and Paulo Quaresma. A Content-Aware Hybrid Architecture for Answering Questions from Open-domain Texts. In *19th International Conference on Computer and Information Technology*, 2016.
- 22 Xiangzhou Huang, Baogang Wei, and Yin Zhang. Automatic Question-Answering Based on Wikipedia Data Extraction. In *10th International Conference on Intelligent Systems and Knowledge Engineering, {ISKE} 2015, Taipei, Taiwan, November 24-27, 2015*, pages 314–317, 2015. doi:10.1109/ISKE.2015.78.
- 23 Inc., Wolfram Research. Wolfram Alpha, 2018.
- 24 Aditya Jain, Gandhar Kulkarni, and Vraj Shah. Natural language processing. *International Journal of Computer Sciences and Engineering*, 2018.
- 25 Michael Kaisser and Tilman Becker. Question Answering by Searching Large Corpora With Linguistic Methods. In *TREC*, 2004.

- 26 S. Kalaivani and K. Duraiswamy. Comparison of question answering systems based on ontology and semantic web in different environment. *Journal of Computer Science*, 8(8):1407–1413, 2012. doi:10.3844/jcssp.2012.1407.1413.
- 27 Jinhuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint*, 2019. arXiv:1901.08746.
- 28 Sweta P. Lende and M. M. Raghuwanshi. Question answering system on education acts using NLP techniques. In *IEEE WCTFTR 2016 - Proceedings of 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare*, 2016. doi:10.1109/STARTUP.2016.7583963.
- 29 Marjan Mernik, Jan Heering, and Anthony M Sloane. When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, 37(4):316–344, 2005.
- 30 V. A. Mochalova, Kuznetsov V. A., Mochalov V., and A. Ontological-semantic text analysis and the question answering system using data from ontology. *ICACT Transactions on Advanced Communications Technology (TACT) Vol.*, 4(4):651–658, 2015.
- 31 FJ Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, volume 1, pages 160–167, 2003. doi:10.3115/1075096.1075117.
- 32 Terence Parr. *The Definitive ANTLR 4 Reference*. Pragmatic Bookshelf, 2nd edition, 2013.
- 33 Warren J. Plath. Request: A natural language question-answering system. *IBM Journal of Research and Development*, 20(4):326–335, 1976.
- 34 P Selvi Rajendran and Rufina Sharon. Dynamic question answering system based on ontology. In *2017 International Conference on Soft Computing and its Engineering Applications (icSoftComp)*, pages 1–6. IEEE, December 2017. doi:10.1109/ICSOFTCOMP.2017.8280094.
- 35 Marcos Ramos, Maria João Varanda Pereira, and Pedro Rangel Henriques. A {QA} System for learning Python. In *Communication Papers of the 2017 Federated Conference on Computer Science and Information Systems, FedCSIS 2017, Prague, Czech Republic, September 3-6, 2017.*, pages 157–164, 2017. doi:10.15439/2017F157.
- 36 Unmesh Sasikumar and L Sindhu. A Survey of Natural Language Question Answering System. *International Journal of Computer Applications*, 108(15), 2014.
- 37 William Stallings. *Cryptography and network security: principles and practice*. Pearson Upper Saddle River, 2017.
- 38 Maria Vargas-Vera and Miltiadis D Lytras. Aqua: A closed-domain question answering system. *Information Systems Management*, 27(3):217–225, 2010.
- 39 David L Waltz. An english language question answering system for a large relational database. *Communications of the ACM*, 21(7):526–539, 1978.
- 40 Dirk Weissenborn, Georg Wiese, and Laura Seiffe. FastQA: A simple and efficient neural architecture for question answering. *arXiv preprint*, 2017. arXiv:1703.04816.
- 41 Tatu Ylonen. Ssh-secure login connections over the internet. In *Proceedings of the 6th USENIX Security Symposium*, volume 37, 1996.