Automated Elicitation of Functional User Requirements for Supporting Cloud Service Search

Andrea Horch, Constantin Christmann and Holger Kett
Fraunhofer Institute for Industrial Engineering IAO
Stuttgart, Germany
{andrea.horch, constantin.christmann, holger.kett}@iao.fraunhofer.de

Abstract—**The use of cloud services can generate tremendous benefits for companies, especially for small and medium-sized enterprises (SMEs). When searching for a cloud service the specification of the required functions is an important aspect for finding appropriate services. SMEs which may not have access to the knowledge of an information technology (IT) expert can have difficulties in formulating their functional requirements and current cloud service search engines do not provide comfortable assistance for the specification of functional requirements like service features. The contribution of this paper is a technique for the automated identification of service features for supporting cloud service search. The technique uses an ontology to perform an analysis of the functions provided by cloud services or on-premise software which is currently used by the SMEs. The technique was implemented and evaluated in the context of the crafts domain.**

Keywords–Functional Requirements; Requirements Elicitation; Cloud Services; Service Search.

I. INTRODUCTION

The adoption of cloud computing holds vast potential for companies. In particular small and medium-sized enterprises (SMEs) can exploit the benefits of cloud computing like such as the outsourcing of resources through the use of a completely external infrastructure or the improvement of processes by using readily available services, which can be accessed from anywhere [11]. Due to its numerous advantages cloud computing already plays an important role for business. Moreover, for small and medium-sized enterprises (SMEs) which may not dispose of high IT expertise it can be difficult to formulate their functional requirements which are an important aspect when searching for suitable software tools. Hence, there is a need to assist the SMEs during the process of functional requirements elicitation when searching for appropriate cloud services.

Current cloud service registries and search engine projects like Cloud Search Portal [3] or the service directories of cloudbook [2] and Cloud Showplace [13] are not adequate in supporting the elicitation of functional requirements during the search process (see Section II). This paper contributes a technique to automatically identify required software features, which is based on an ontology. The ontology represents the domain of software features in the context of the crafts sector. The method was implemented in form of a search engine for the identification of required cloud service features in the crafts domain. The evaluation has verified the suitability of the technique for supporting users during the process of the identification of required cloud service features for cloud service search. The paper is organized as follows: Section II presents related work and Section III is introducing the approach. The implementation of the technique is presented in Section IV. The approach of the evaluation and the results are shown in Section V. We conclude in Section VI and we propose the future work in Section VII.

II. RELATED WORK

Table I gives an overview of previous projects in the field of cloud service search and the tasks they focus on. The cloud portal introduced in [5] implements as a main feature a cloud service search engine. The search engine is based on a domain ontology describing the main features of a cloud service like service type, vendor details, technical information or cost and time details. Cloud service providers use the terms of the ontology to describe their services. Additionally, they have the possibility to add some keywords to the service description. Service consumers, which are searching for an appropriate service, also use the terms of the ontology to describe their requirements for service search. Furthermore, they have the option to add keyword for the search. The matching is done by similarity reasoning based on the domain ontology. The cloud service search engine CloudRecommender presented in [14] is based on an ontology called Cloud Computing Ontology (CoCoOn). The project focuses on the identification and selection of infrastructure services, which belong to the infrastructure as a service (IaaS) model of cloud computing. Users can search by selecting basic configuration parameters for the searched cloud service, which are provided by the domain ontology. Additionally, regular expressions can be used to search for appropriate services. The domain ontology designed in [6] describes cloud services. The ontology is used to assist cloud service providers to describe the cloud services when registering a cloud service in a cloud service registry. The ontology acts as a standard terminology for the service descriptions of the different providers. They have implemented a keyword-based cloud service search engine, which matches the keyword query to the terms of the ontology and infers suitable services. In [7], Cloud Service Crawler Engine (CSCE) is presented which is a crawler engine for cloud service discovery and categorization. CSCE crawls the internet by using the application programming interfaces (APIs) of conventional search engines like Google, Bing or Yandex, as well as a domain ontology describing cloud service characteristics. The crawler uses the concepts in the first levels of the ontology as query keywords to feed the search engines and the `is-a` and `is-not-a` relations between the concepts for the decision whether a search engine result is a cloud service or not. The ontology is also used for the categorization of the service providers of the discovered cloud services into infrastructure as a service (IaaS), platform as a service (PaaS), software as a service (SaaS), IaaS+PaaS, IaaS+SaaS, PaaS+SaaS and all.

The contribution of [10] is an approach for the generation of semantic cloud service descriptions. The proposed technique transforms a natural language description of a cloud service into a semantic service description by using a domain ontology, as well as natural language processing (NLP) tools. Based on the semantic description of the cloud services a search index is generated, which is used for processing a keyword-based search.

There are further recent approaches which pursue the goal to ease the use and reuse of cloud services. One of those projects is the Artist (Advanced software-based seRvices provisioning and migraTIon of legacy SofTware) project [1] which offers a set of methodologies and tools to owners and developers of cloud services for transforming their software in a way that it can exploit the benefits of cloud features. Another project called REMICS (REuse and MIgration of legacy applications to Interoperable Cloud Services) [9] directly translates natural user requirements to Java code by transforming natural language phrases into the Requirements Specification Language (RSL). Since the approach of the Artist project needs a deeper IT know-how and thus, is more suitable for owners and developers of cloud services than for non-experts it will be not further considered in this paper. The approach of the REMICS project offers the possibility to the users to easily define their software requirements in natural language, but it does not give them a hint if they do not completely know their requirements. As our approach aims to assist the users in identifying their functional requirements it needs to use other techniques than proposed in [1] and [9]. [1] and [9] are not compared to the cloud service search approaches in Table I since they do not mainly focus on cloud service search.

As illustrated in Table I the previous projects presented in this section deliver solutions for cloud service discovery, the description of cloud services as well as the elicitation of non-functional user requirements for cloud services. The elicitation process for the required features is only partially solved by some projects. These projects, which deliver a partial solution for the investigation of the required service features only consider the service type like IaaS or some technical requirements like the available storage size, but they do not consider the denotations of functions nor determine the functions offered by a service. Thus, there is a deficit in the assistance of users during the elicitation process of their required service features when searching for appropriate cloud services, which is filled by the technique introduced in this paper.

III. Approach

Our approach for an automated identification of required service features is based on the assumption that the users already have a software in use, which they want to substitute with an appropriate cloud service. Another possible scenario is that the user knows an on-premise software having all required functions and they are looking for a corresponding cloud service. Figure 1 shows the steps of our approach for automated elicitation of required service features.

In the first step, the user enters the search query, which is the URI of the software website, which should be substituted by the cloud service, or the URI of a similar software having the required functions. Having received the website URI, the content of the website and of all pages linked on it have to be crawled. In the next step, the content is checked against a domain ontology, which contains software function classes. These classes include several labels of function terms. Each class of the ontology is checked for its occurrence in the crawled content. Classes whose labels match a term in the content, are added to the result set. After that step the result set is transformed into a hierarchical order corresponding to the hierarchy of the function classes in the domain ontology. Finally, the hierarchical result list of software functions is displayed to the user.

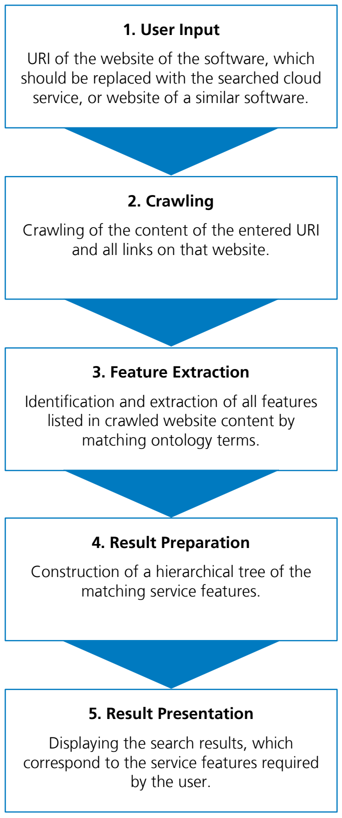

1. User Input
URI of the website of the software, which should be replaced with the searched cloud service, or website of a similar software.
2. Crawling
Crawling of the content of the entered URI and all links on that website.
3. Feature Extraction
Identification and extraction of all features listed in crawled website content by matching ontology terms.
4. Result Preparation
Construction of a hierarchical tree of the matching service features.
5. Result Presentation
Displaying the search results, which correspond to the service features required by the user.

Figure 1. Approach for automated elicitation of required service features

IV. Implementation

The implemented system consists of a user interface, a domain ontology, a component for web content crawling, a component for feature extracting and a component for processing the results.

A. Ontology Creation

For the implementation of the approach a domain ontology is needed for identifying terms of functions and interfaces in the crawled content of the software websites. For the creation of the ontology we used the ontology editor Protégé [12] and followed the Ontology Development 101 methodology described in [8]. The Ontology Development 101 methodology comprises the following steps:

1) Determination of domain and scope of the ontology.

TABLE I. OVERVIEW OF CLOUD SERVICE SEARCH PROJECTS

	cloud service discovery	generation of cloud service description	identification of required functions	identification of other features
[5] J. Kang and K. M. Sim, 2011	●	●	◐	●
[14] M. Zhang et al., 2013	○	●	◐	●
[6] V. S. K. Nagireddi and S. Mishra, 2013	○	●	○	○
[7] T. H. Noor et al., 2013	●	○	○	○
[10] R. Sahandi, A. Alkhalil and J. Opara-Martins, 2013	○	●	○	○

Legend: ○ no solution, ◐ partial solution, ● solution

2) Consideration of reusing existing ontologies.
3) Enumeration of important ontology terms.
4) Definition of the classes and class hierarchy.
5) Definition of the properties of classes (slots).
6) Definition of the facets of the properties (cardinality, value type, domain and range).
7) Creation of instances.

The domain of the ontology covers functional cloud service requirements such as functions and interfaces. It shall be used for the elicitation of the functional requirements by analysing the functional range of a currently used software tool or a similar software from the software description on the website of the provider. We restricted the ontology to describe cloud software in the crafts domain, but it was designed to be easily extended to other domains. Concerning step 2 of the methodology, we have analysed the ontologies of previous work whether these can fit our needs. For this purpose we examined the ontologies used in [4]- [7]. All these ontologies offer several meta data for describing software, especially cloud software. However, they do not include terms or a hierarchy for describing the concrete functions and interfaces of software. Hence, we decided to create a new ontology which includes terms and a hierarchical structure of software functions and interfaces. Important ontology terms where identified by scanning the websites of five different software services of the crafts domain for terms of functions and interfaces and adding them to the ontology in a hierarchical structure. The ontology consists of a class hierarchy (`is-a` relationships) and different labels for the classes to express synonyms.

Since we plan to extend the ontology to store additional knowledge about the data which is processed by each function in order to be able to increase our system for the search for trusted cloud services it was created in the Resource Description Framework Schema (RDFS) which is a standard of the W3C. For the search for trusted cloud services it is important to know if a service processes or stores critical data like personal data of customers or business secrets.

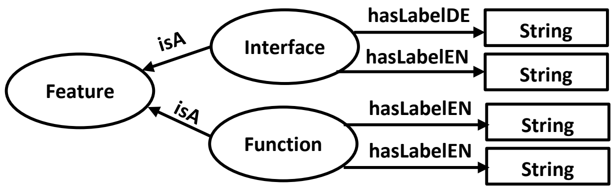


Figure 2. Structure of the software feature ontology

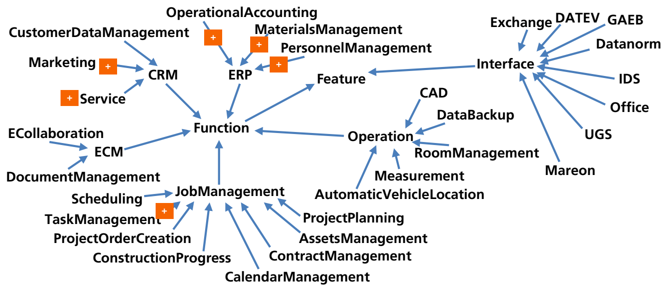


Figure 3. Extract of the software feature ontology

Figure 2 presents the structure of the ontology, whereas Figure 3 shows an extract of the ontology. The class `feature` describes all features of a software and includes the subclasses `function` and `interfaces`, which are features of a software. The class `function` contains several subclasses, which represent concrete functions of a cloud software in the crafts domain like Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM) and their subclasses. The class `interface` is a meta class of software interfaces including several subclasses describing concrete interfaces of cloud software in the crafts domain like an office interface or a DATEV interface for transferring product data.

Moreover, the ontology contains the meta classes `FunctionMetaclass` for the description of software functions and a meta class `InterfaceMetaclass` for software interfaces. Another meta class is an extension of the system class `Slot` called `SlotWithLabel` which was designed to be able to add multilingual labels in the form of class properties to a function and an interface class. The ontologies includes the slots (class properties) `labelDE` and `labelEN` which both have the domains `Function`, `FunctionMetaclass`, `Interface`, `InterfaceMetaclass` and `SlotWithLabel`. The properties use the value type `String` for adding English and German labels to a function or interface class in order to define several English and German synonym terms for describing the name of a service function or interface. The ontology includes 36 classes for service functions and 10 different interface classes. It consists 118 German labels and 111 English labels for describing the different function classes. It has 16 German labels as well as 16 English labels for naming the interface classes.

B. System Built

After building the ontology the system for the identification and extraction of required service functions was created. Figure

4 presents the system design.

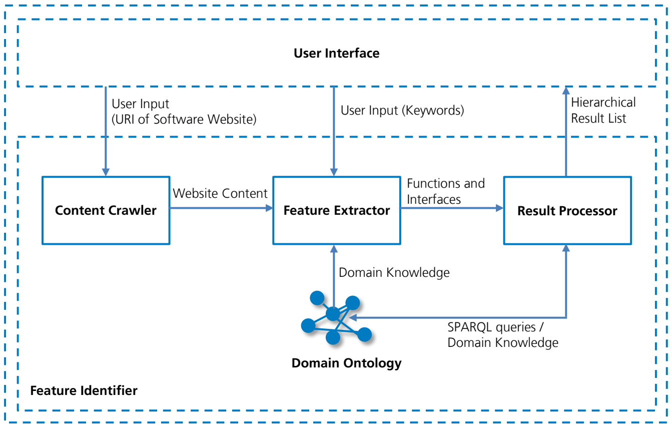
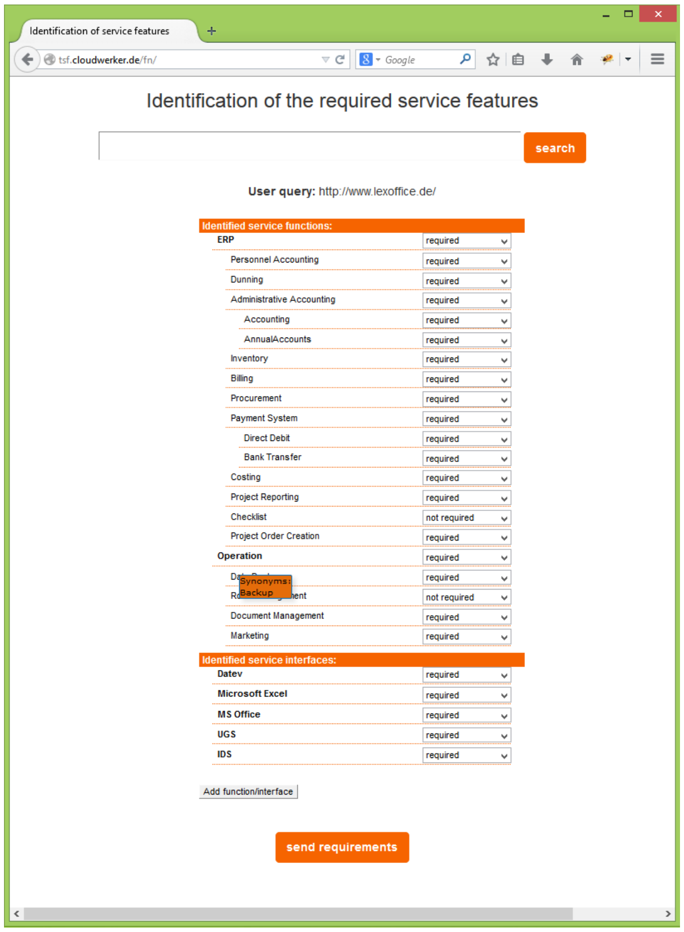


Figure 4. System design

- **User Interface:** The *user interface* is a web application where the users can enter a search query in form of a URI of the software to substitute or a software, which offers a similar set of functions and interfaces as the searched service. We designed the system to use the online description of an actually used software (on-premise software or web service) on the website of the software provider or retailer since we assume that the user will already use a software.
  Having processed the search the results are presented in the *user interface* in the form of hierarchical lists of the identified functions and interfaces. The users can decide, which of the listed features they require and send their selection to the service search engine. An example of displayed results is shown in Figure 5.
- **Content Crawler:** If the user query is the URI of the website the *content crawler* crawls the content of that website and all websites of the same domain, which are linked on it. After the crawling process the *content crawler* transfers the crawled content to the *feature extractor*.
- **Feature Extractor:** The *feature extractor* uses the *domain ontology* to compare the class labels (terms of functions and interfaces) with the website content. All matching ontology classes are added to the result list, which is referred to the *result processor*. Since the crawling is done in real-time during the search operation and some website linking many other domain websites we decided to interrupt the crawling after three minutes. That means, the functions and interfaces only occurring on websites, which are not crawled within three minutes, will not be identified by the system. The interruption after a time of three minutes was chosen because a three minutes time space for crawling has provided good results in our tests - even for bigger websites. We plan to reduce the processing time of three minutes for later versions of the system (e.g. by incremental result generation or a parallel programming approach).
  If the user query is formulated as keywords the *feature extractor* checks the ontology classes directly against the keywords.
- **Result Processor:** The *result processor* takes the result list from *feature extractor* and prepares it for the presentation on *user interface*. It processes SPARQL Protocol And RDF Query Language (SPARQL) queries on the *domain ontology* to get information about the hierarchical structure of the results and converts the result list to hierarchical lists of functions and interfaces, which are transmitted to the *user interface*, which is shown in Figure 5.



Identification of the required service features

User query: http://www.lexoffice.de/

Identified service functions:	
ERP	required
Personnel Accounting	required
Dunning	required
Administrative Accounting	required
Accounting	required
AnnualAccounts	required
Inventory	required
Billing	required
Procurement	required
Payment System	required
Direct Debit	required
Bank Transfer	required
Costing	required
Project Reporting	required
Checklist	not required
Project Order Creation	required
Operation	required
Da[illegible]	required
R[illegible]ent	not required
Document Management	required
Marketing	required

Identified service interfaces:	
Datev	required
Microsoft Excel	required
MS Office	required
UGS	required
IDS	required

Figure 5. Result lists of the identification of required service features

Figure 5 shows an example for a result list from the system for a search for suitable functional service requirements which can be used for a cloud service search in order to substitute a currently used service.

V. EVALUATION

The evaluation of our approach is separated into two parts. In the first part, we measure the adequacy of the approach in eliciting the functional user requirements from software websites for being able to run a precise search for relevant cloud services. The second part of the evaluation validates the results of the elicitated functional requirements for running a cloud service search.

TABLE II. CONSTRUCTION WEBSITES

software	URL
pds abacus	http://www.pds.de/cms/produkte/pds-abacus/
myfactory	http://www.myfactory.com/
scopevisio	https://www.scopevisio.com/
moser	http://www.moser.de/produkte/mosaik/
kwp-bnWin.net	http://www.kwp-info.de/

TABLE III. CONTROL WEBSITES

software	URL
lexoffice	http://www.lexoffice.de/
midcom	http://www.midcom.de/
cas pia	http://www.cas-pia.de/
reporta	http://reporta.ag/de/
salesking	http://www.salesking.eu/

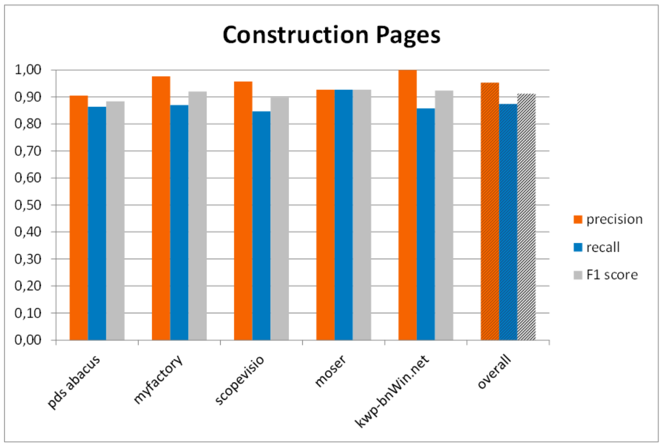


Figure 6. Precision, recall and F1 score for the construction pages

A. Elicitation of Functional Service Requirements

The first part of the evaluation analyses the quality of our system for the elicitation of the functional user requirements from software websites by measuring precision, recall and F1 score for the identification of the functions and interfaces offered by software systems, which are described on (1) the software websites we used to build up the domain ontology (construction pages) and (2) other software websites of the crafts domain (control pages), which act as control group.

Criteria for the selection of the construction and control pages were the following:

- **Construction pages:** The construction pages should contain a description of a cloud service of the crafts domain including as many functions and interfaces as possible.
- **Control pages:** The control pages should also contain a description of a cloud service of the crafts domain. The main criteria for a control page was that the service described should differ a lot in respect to structure and terms from those described in the construction pages.

The websites for ontology construction are given in Table II whereas the Uniform Resource Locators (URLs) for the control websites are quoted in Table III.

For this purpose we manually collected all functions and interfaces described on the mentioned websites and compared these results with the result lists delivered by our system. The evaluation results for the software websites used to build up the ontology are shown in Figure 6.

For the construction pages we could achieve an average value for precision of 0.95. We could reach an overall value for recall of 0.87 and for the F1 score of 0.91. Reasons for not achieving 1.00 for all values are the following:

- Crawling interruption: We interrupt the website crawling during the identification process after three minutes. Functions and interfaces only occurring on web pages, which are not crawled could not be identified.
- Wording on websites: Some software functions are described on the website using a sentence and not a single term of the ontology. There is no match between the terms of the ontology and the text on the website, which describes the function.
- Asynchronous JavaScript and XML (AJAX): Some websites use AJAX for displaying some information. This information is only displayed after triggering a specific action. Since we do not trigger actions during the crawling process functions and interfaces described in text, which only occurs after an action, does not become identified.
- No semantic analyses of content: Our process does not analyse the semantics of the crawled website content. There are sentences like ”You can pay the service fee by bank transfer or direct debit”, which specifies how a customer can pay for the use of the software described on the website. However, the system gets a match between the words ”bank transfer” and ”direct debit” in the sentence on the website and the corresponding ontology classes describing the software functions ”bank transfer” and ”direct debit”.

The results for the control pages are shown in Figure 7.

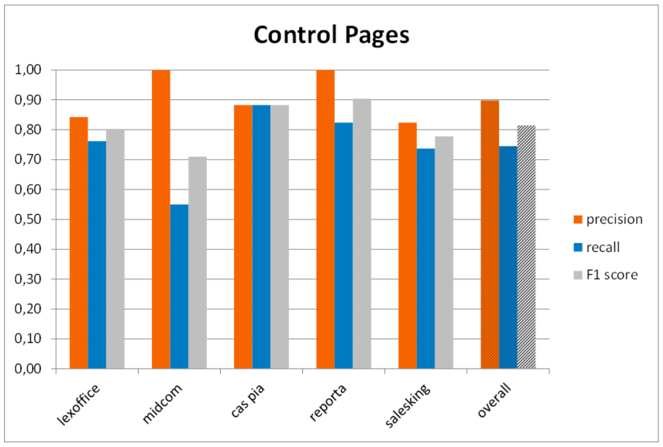


Figure 7. Precision, recall and F1 score for the control pages

For this type of pages we achieved an average value for precision of 0.90, and overall recall of 0.74 and an F1 score of 0.81. This means, on average the system can identify 74 %

of the features listed on software website not used to build the ontology, which is a sufficient result for a system, supporting users to identify their required service features and interfaces.

B. Cloud Service Search

In this part of the evaluation, the suitability of our approach for cloud service search is validated. For this purpose we take the results from the prior step of functional service requirements elicitation for each of the control websites and use the identified service functions and interfaces as input for a cloud service search by using the search engine Google. We used the Google search engine for our approach since the previous approaches described in Section II were not available online. The search query for the cloud service search is created by the leading keyword "cloud service" followed by the required service functions identified in the requirements elicitation step followed by the determined service interfaces. Thus, a search query for the evaluation has the following structure:

"cloud service *<function 1> <function 2> <function 3> ... <interface 1> <interface 2> ...*"

For the service discovery only the search results of the first result page of the Google search were taken into account and maximum one link from search result page to reach the website of an appropriate cloud service was followed. For empty result sets of a keyword search or not sufficient result sets the keyword for a minor interface or function was deleted and a new search was started. For finding the cloud services of Table IV a maximum of three keywords from the results of the requirements elicitation step was deleted.

Table IV presents the results of a Google search for cloud services by using the functional requirements reached by the first step of the approach as search keywords. The table shows the results for five different searches for a substitution of the software services "lexoffice", "midcom", "cas pia", "reporta" and "salesking" by appropriate cloud services. Each search has returned a minimum of three possible services. The results were checked for their suitability to substitute the initial service by comparing the functions and interfaces of the detected service with those of the initial service. In Table IV the suitability for substitution of a detected service is described as "unsuitable" if no or only one function, which is not the main function of the initial service, of the detected service matches the functions of the initial service. The suitability for substitution of a service identified during the search is called "partially suitable" if some functions of the initial service can be covered by the service, but not all. The suitability for substitution of a service is denoted as "suitable" since all functions of the initial service can be substituted by the discovered service. The results show that for each service of the control group at least one cloud service could be found which can totally substitute the initial service. For most services even two or more cloud services for the service replacement could be discovered. The outcome of our evaluation has demonstrated the adequacy of our approach for the automated elicitation of functional requirements to search for appropriate cloud services.

TABLE IV. RESULTS OF THE CLOUD SERVICE SEARCH

Cloud service	**Suitablility for substitution**
lexoffice	
Exact Online http://www.exactonline.de/	●
Microsoft Dynamics http://www.microsoft.com/de-de/dynamics/default.aspx	●
Sage Office Online https://www.sage-office-online.de/	●
midcom	
Salesforce https://www.salesforce.com/	○
umantis Talent Management http://www.umantis.com/	○
karg-edv emis http://www.karg-edv.de/	●
cas pia	
Exact Online http://www.exactonline.de/	●
Salesforce https://www.salesforce.com/	○
SuperOffice CRM Online http://crmonline.superoffice.de/crmonline_062014-1/	●
pds abacus http://www.pds.de/cms/produkte/pds-abacus/uebersicht.html	●
Microsoft Dynamics http://www.kumavision.com/microsoft-dynamics-crm	●
reporta	
Salesforce https://www.salesforce.com/	○
zep http://www.zep.de/	●
LogMyTime http://www.logmytime.de/	◐
Odoo OpenERP http://www.ife.de/	●
TimeNote http://www.timenote.de/	◐
timr http://www.timr.com/	◐
salesking	
Exact Online http://www.exactonline.de/	●
Salesforce https://www.salesforce.com/	○
SuperOffice CRM Online http://crmonline.superoffice.de/crmonline_062014-1/	◐
CRM on Demand http://www.crm-on.de/	◐
projectfacts https://www.projectfacts.de/	●

Legend: ○ unsuitable, ◐ partially suitable, ● suitable

VI. Conclusion

The contribution of this paper is a technique for the automated identification of required (cloud) service functions. We have implemented the approach in a system for the automated identification and extraction of required cloud service functions in the crafts domain. The system is based on a domain ontology, which was manually built up by collecting the functions listed on the websites of several cloud services for the crafts domain. We have evaluated the system by measuring its precision, recall and F1 score according to two different test cases. In the first test case the system has identified the functions and interfaces listed on the websites the domain ontology was built up from. In the second case the system had to examine the functions and interfaces from websites of other crafts software as control group. For the control group websites, we have got an average value for recall of

0.74. This evaluation result is sufficient for a system, which supports users to identify required service features. Moreover, we have performed a cloud service search by using the elicited functional service requirements as keywords for running a Google search. The results of the Google searches for the control group services have returned adequate cloud services. Thus, we could demonstrate the suitability of our approach for functional requirements elicitation to support searches for suitable cloud services.

VII. Future Work

There are some ideas which can improve our approach and which are planned for future work. Currently, the users can only enter an URI or keywords into the system for identifying required (cloud) service features. We plan to offer the possibility to process a keyword search after having received the results for an URI-based search to provide an iterative search process to the users. Another idea is the possibility to search for the required features of several software. That means the users can enter the URIs of multiple software websites to get the all functions and interfaces offered by all software tools together. An additional consideration is to extend the domain ontology automatically by software descriptions of a repository or by user input. The case of extending the ontology based on user input could be implemented into the user interface. A possibility could be to implement a dynamic result where the users can add own classes of functions and interfaces in form of terms into the hierarchy of the result list. Thus, the terms entered by the users could be automatically added into the hierarchy of the domain ontology. As mentioned in Section IV-B we plan to reduce the waiting time for the users e.g. by incremental result generation or a parallel programming approach. The final version of the system shall be evaluated for its practical usage by running a user study.

Acknowledgement

This work was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference “01MD11041”. The authors take the responsibility for the contents.

References

[1] Athens Technology Center, “Artist Project Website”, 2015 retrieved on March 8, 2015 from http://www.artist-project.eu/.

[2] The Cloudbook Community, “cloudbook.net”, 2013 retrieved on March 8, 2015 from http://www.cloudbook.net/directories/product-services/cloud-computing-directory.

[3] CloudSearchPortal, “CloudSearchPortal.com”, 2015 retrieved on March 8, 2015 from http://www.cloudsearchportal.com/.

[4] H.-J. Happel , A. Korthaus, S. Seedorf and P. Tomczyk, “KOntoR: An Ontology-enabled Approach to Software Reuse”, Proceedings of the 18th International Conference on Software Engineering and Knowledge Engineering, 2006, pp. 349-354.

[5] J. Kang and K. M. Sim, “A Cloud Portal With a Cloud Service Search Engine”, Proceedings of the 2011 International Conference on Information and Intelligent Computing IPCSIT, vol. 18, IACSIT Press, Singapore, 2011, pp. 1-8.

[6] V. S. K. Nagireddi and S. Mishra, “An Ontology Based Cloud Service Generic Search Engine”, Proceedings of the 8th International Conference on Computer Science & Education (ICCSE), 2013, pp. 335-340.

[7] T. H. Noor, Q. Z. Sheng, A. Alfazi, A. H. H. Ngu and J. Law, “CSCE: A Crawler Engine for Cloud Services Discovery on the World Wide Web”, Proceedings of the 20th International Conference on Web Services (ICWS), 2013, pp. 443-450.

[8] N. F. Noy and D. L. Mcguinness, “Ontology Development 101: A Guide to Creating Your First Ontology”, tech. report, Stanford University, 2001.

[9] REMICS, “REMICS Project Website, Reuse and Migration of legacy applications to Interoperable Cloud Services”, 2011 retrieved on March 8, 2015 from http://www.remics.eu/home.

[10] M. Á. Rodríguez-García et al., “Semantic Annotation and Retrieval of Services in the Cloud”, Distributed Computing and Artificial Intelligence, vol. 217, Springer International Publishing, 2013, pp. 69-77.

[11] R. Sahandi, A. Alkhalil and J. Opara-Martins, “Cloud Computing From SMEs Perspective: A Survey-based Investigation”, Journal of Information Technology Management, vol. 24, No. 1, University of Baltimore, 2013, pp. 43-49.

[12] Stanford Center for Biomedical Informatics Research, Stanford University, “Protégé”, 2014 retrieved on March 8, 2015 from http://protege.stanford.edu/products.php.

[13] THINKstrategies, “Cloud Computing Showplace”, 2012 retrieved on March 8, 2015 from http://www.cloudshowplace.com/.

[14] M. Zhang et al., “Investigating Techniques for Automating the Selection of Cloud Infrastructure Services”, International Journal of Next-Generation Computing, vol. 4, no. 3, 2013, pp. 759-764.