Automatic Service Retrieval in Converged Environments Based on Natural Language Request

Edgar Camilo Pedraza
GIT
University of Cauca
Popayán, Colombia
epedraza@unicauca.edu.co

Julián Andrés Zúñiga
GIT
University of Cauca
Popayán, Colombia
gzja@unicauca.edu.co

Luis Javier Suarez Meza
GIT
University of Cauca
Popayán, Colombia
ljsuarez@unicauca.edu.co

Juan Carlos Corrales
GIT
University of Cauca
Popayán, Colombia
jcorral@unicauca.edu.co

***Abstract*—Finding services in dynamic and heterogeneous contexts, as converged environments (Next Generation Networks), is a very complex task and a crucial aspect in this new paradigm of convergence. Therefore, it is essential to have efficient and effective mechanisms for seeking services, to take advantage of resources in the network (Web and telecommunications services). Recent studies have developed Service Creation Environments for Telecommunications and Internet converged services, where user´s requests are represented by complex expressions that describe the required services. Thus, the search and selection of these services depend on the ability of the developer to retrieve the most suitable ones, converting this labor in an inefficient work. With this in mind, and in order to improve the time to create convergent services, this paper proposes a novel approach that supports the automatic retrieval of services in converged environments, considering functional and non-functional requirements of end-user's requests in natural language, to optimize the process of convergent services creation.**

***Keywords-automatic service retrieval; converged environments; natural language request; Telecom and Internet converged service*.**

I. Introduction

The ability to retrieve services that accomplish user requests, has led to the development of various projects, focused on service retrieval, understood as an important stage in the composition process [1] that allows the user to find and use a service based on a published description of its functionality or operational parameters [2]. Currently, services retrieval offers new challenges driven by the convergence of Web and telecommunications domains around the IP protocol, enabling the use of diverse and innovative services, regardless of the customer access network [3]. The above-mentioned, both with new trends in application environments, where users are important generators of content and applications, opens up towards a new paradigm in which non-technical individuals are able to design and create their own fully customized services by integrating Web and telecommunication components, an activity that years ago was done only by expert developers due to its complexity.

From the above notion, in this paper, we propose an architecture for automatic service retrieval in converged environments, considering the services functional properties (e.g., inputs, outputs, preconditions and effects) and nonfunctional properties (e.g., QoS, such as: availability, response time, reputation, etc) requested by the user in natural language (NL), to speed up the creation of converged services.

Typically, natural language processing (NLP) is useful to analyze and produce semantic representations of the user's request, providing information needed to identify the generic control flow, as well as functional and nonfunctional properties of services. Therefore, in this paper, we propose the use of NLP techniques to automate the process of service retrieval from requests made in NL. Thus, with semantic descriptions supported by NLP techniques and adaptation of algorithms for matching services and user's request, it is possible to make an accurate and automatic retrieval of services available within both Web and telecommunications domains.

The remainder of this paper is structured as follows: in the next section, we review the work related to the different topics involving the current research. Then, in Section III, a high level description of the proposed architecture is presented. To provide greater clarity an example is discussed in Section IV. Finally, in Section V we conclude the paper.

II. Related work

Service retrieval can be addressed under two main approaches: syntactic and semantic. The searching of services from a syntactic approach, considers either, interfaces matching techniques or keyword searching [4] that require exact matches at the syntactic level between the descriptions of services and the parameters used, which leads to deficient results in the retrieval of service. The semantic approach allows the establishment of relationships between concepts that define the functionality of services (functional properties) and additionally, considers formal descriptions constructed by non-functional properties [5], achieving a more precise description of services, improving the quality of results to retrieve services according to user needs [6].

From the foregoing, and given the nature of the problem, the proposed solution focuses on services retrieval based on semantics and NLP. In this sense, some existing solutions are described below. In [7], the IBM research team developed a supercomputer that performs analysis phases of natural language questions composed by hundreds of algorithms, some of these phases present a similar approach with the proposed ones in this paper. However, it requires a complex system composed of multi-core hardware processor.

In [8], the authors address the selection of Web services based on requests expressed in NL, this solution is based on language restrictions, matching the structure of the request with predefined patterns for decomposition into blocks using keywords. Subsequently, the request is processed and transformed into a data flow and control model expressing the general logic of the new service. In addition, the authors propose the use of a common ontology and a NL dictionary, in order to relate textual fragments with functional parameters of such services. This paper presents disadvantages when limiting requests to simple sentences.

Based on concepts graphs and conceptual distance measure, a solution is presented in [1]. Its purpose is to calculate the similarity between the user's request, represented by keywords, and services available in a repository. Within the linguistic analysis, different processes are performed: text segmentation, irrelevant word removal, elimination of derivatives (stemming) and grammatical corrections. The authors admit their proposal's lack of dynamic adaptation at runtime.

The approach of [5] re-uses the converged services creation environment of project SPICE (Service Platform for Innovative Communication Environment) [9, 10] to facilitate services retrieval with different types of semantic annotations. The author focuses his work on the development of an intelligent agent in charge of analyzing the application in NL to extract semantic information, specifically the goals, from which additional semantic information is derived, as inputs and outputs, which are used to retrieve services and report their order composition. However, retrieval's throughput and processing critically depend on the amount of services stored in the repository.

From the previous review, limitations are evident because they are based on functional preferences, leaving aside non-functional requirements that provide great sense of services semantic descriptions. Finally, the automatic retrieval of services in converged environments is a recent topic of research, where, considering the above characteristics, no work has been done.

III. ARCHITECTURE

In this section, we present our proposed architecture for automatic services retrieval in converged environments, which receives as input the user's request made in NL from a mobile device, and gives a service ranking and a generic control flow, as output. Figure 1 shows the modules of the architecture, organized in four phases, which correspond to: *Natural Language Analysis, Matching, Recommender and Inference phase*, these modules are described below:

- *Tokenizer*: as input, it has the NL request and from this, it obtains words, phrases or symbols called tokens.
- *Filter Words*: responsible for removing non-sense words by comparing with a set of words previously identified.
- *Words Tagging*: tags words according to its grammatical category (e.g., she "pronoun”, loves "verb", animals "noun").
- *Named Entity Recognition*: classifies the words into “functional” or “control” categories.
- *Semantic Analyzer*: in charge of semantic disambiguation process of input words.
- *Non Functional Requirements Recommender:* searches non-functional parameters in the repository from functional request previously written by user.
- *Services Recommender:* searches request-service information in the repository obtained from prior inputs of users.
- *Matcher Functional Requirements*: obtains from cluster services, the first rank, by matching functional requirements.
- *Ranking Generator*: obtains the final ranking of services considering, if exists, non-functional requirements, of services, such as QoS.
- *Cluster of Services*: conformed by abstract descriptions of Web and Telecommunications domain services, which are conceptually organized as functional properties.
- *Upper Ontology:* involves general concepts that are the same across all knowledge domains (e.g. QoS, Telco, IT, among many others), supporting the functional and non-functional properties description and enabling the ontology reasoning.
- *Flow Ranking Repository:* stores an association of service (tags) obtained at the end of the semantic matching phase.
- *Non functional Repository:* stores an association of non-functional and functional parameters of cluster´s services.
- *Generic Flow Generator*: generates an approximate generic control flow, based on keywords taken from the user request processed.
- *Flow-Ranking Associator:* associates the flow obtained in the *Generic Flow Generator* module with the ranking output of the matching semantic phase, obtaining the final output of the architecture.

Most relationships between architecture modules are sequential. However, there are interactions between stages that do not follow this behavior. Below, a more detailed description is made, of the different phases and processes that take place inside of them.

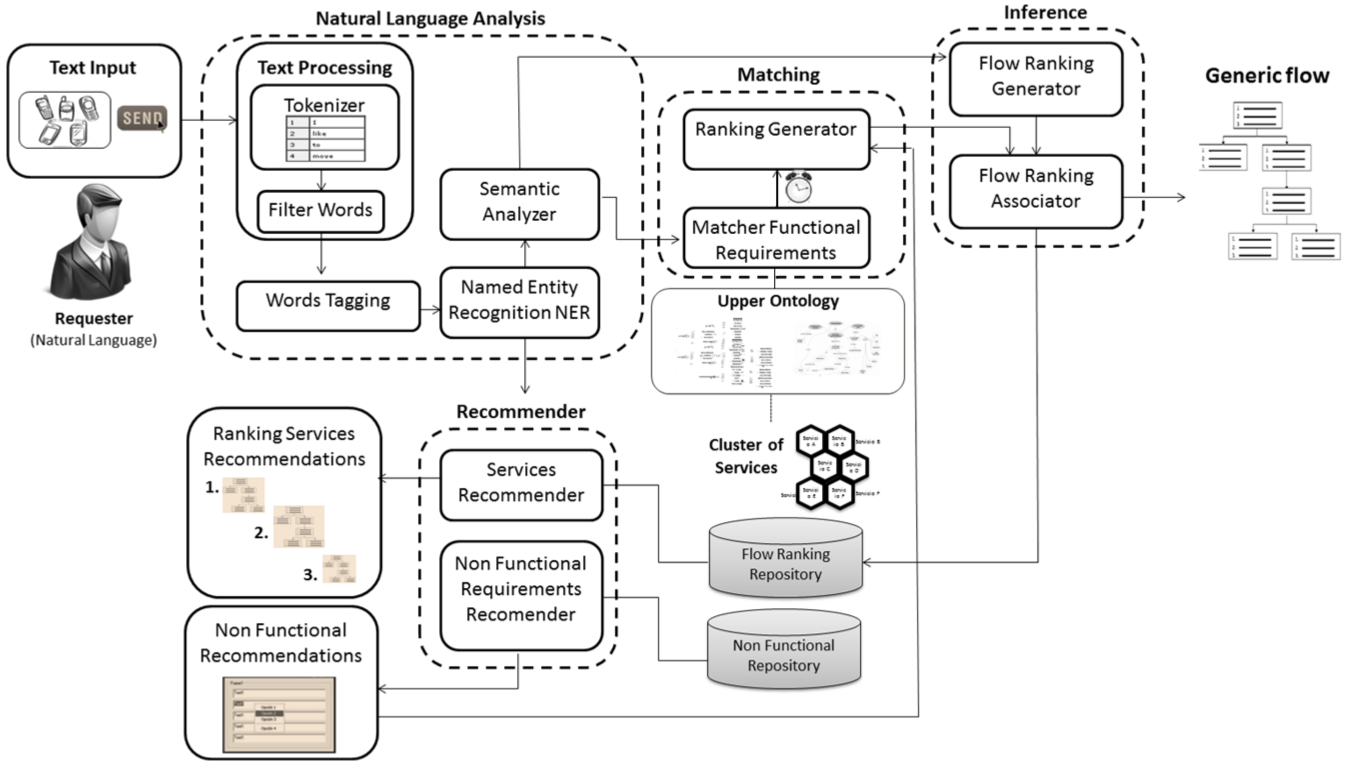

Figure 1. Architecture of the System.

A. Phase of Natural Language Analysis (NLA)

Initially, a user makes a request from his/her mobile device in NL, which is received by the *Tokenizer*, where tokenization operation starts, it obtains simple lexical units from complex sentences, by removing existing spaces. Additionally, this module corrects simple lexical errors that may arise in the request, i.e., misspelled word errors that are easily identifiable. Afterward, the sentences are processed through *Filter Words Module* and later they pass through *Words Tagging*, with which, it is pretended to classify (tag) words of the sentence according to their grammatical category. The module also aims to undertake an analysis based on linguistic rules, trying to identify and compensate syntactic and structural errors. Some techniques used to implement the modules above are GateNLP, OpenNLP, Apache UIMA, among others. One of the most important is GateNLP [12], which offers an architecture that contains functionality for plugging in all kinds of NLP software: (POS taggers, sentence splitters, named entity recognizers) and all are java based.

Once completed these operations, the request is more consistent, but remains complex. Therefore, we established the *Named Entity Recognition*, which performs a classification between "*Control"* and *"Functional"* words according to its meaning, from which, control words are directed to Flow Ranking Generator Module, whereas functional words are directed to *Semantic Analyzer* and also to *Recommender modules*. Within the final stage of linguistic analysis, it is important to consider the semantic ambiguity, for which the *Semantic Analyzer* identifies the correct sense of words according to their context, i.e., identifies the correct one from a word within multiple meanings that can occur in a sentence. This allows an easy identification of keywords with their respective grammatical category (e.g., noun, verb, adjective, conjunction) that define the user's request and with which service selection will be made. Thus, this stage offers the user greater flexibility in the use of language and allows the establishment of a wider range of possibilities. On the other hand, this phase also allows conditional words identification (e.g., *if*, *then*, *later*) and words of order (sequence) (e.g., *first*, *second*) important for the *Generic Flow Generator* in the inference phase.

B. Recommender

This phase is composed of two modules and it is executed while matching phase performs the searched of services with functional requirements. Additionally, the phase shows a hint to the user of generic control flow stored in a repository, if the user select one, the execution of the two remaining phases would be avoided, and straightaway the process finished, otherwise, the process continues in the matching phase. For the above task, the module in charged is the *Service Recommender*, which obtains a list of generic control flow with services of the repository from obtained keywords classified as functional. The second one module is named *Non Functional Requirements Recommender*, this searches non-functional parameters from the *Non*

Functional Repository using the obtained keywords classified as functional, the result is shown to the user, and if he/she selects one, it becomes the input of the *Ranking Generator* module.

C. Matching Phase

At this stage there are two important modules: the first one is related with the use of an upper ontology for the matching of functional requirements, to obtain the first ranking of services (*Matcher Functional Requirements*). The second one refers to the Generator Ranking module which use a *weighting algorithm*, once the first ranking of services is obtained, it weights ranking with non-functional keywords which are taken from the Recommender phase, for which, the *Generator Ranking* module uses a timer. It provides a time out for an entry of those parameters. If after a certain time, income of non-functional words does not exist, the module will get a ranking with only functional parameters. Non-functional parameters are very important in the request because they represent aspects such as quality, efficiency, availability, etc., with which is possible to offer more adequate services to users. It considers that request can be enriched with non-functional parameters, in order to provide optimum results that best fit to end user requirements.

D. Inference phase

This stage begins with the *Generic Flow Generator*, which receives as input Keywords classified as Control obtained from NER through NLA phase. With this, the module infers a basic structure of ordered operations that represents the basic control flow, useful for the composition of services, which is performed after the service retrieval. *Flow-Ranking Associator* receives as inputs a service ranking from the semantic matching phase and the basic control flow (obtained from the previous module) in order to generate the services generic control flow that is stored in *Flow Ranking Repository* and becomes the output of the whole architectures.

IV. EXAMPLE

This section describes the functionality of the proposed architecture through an example that details each of the phases of the process for automatic retrieval of services in converged environments. To do so, consider the following situation. Using natural language, an executive requests from his cell phone a meetings coordination service, so: *"I want to receive traffic reports of Bogotá via messages, minutes before the meeting and if I have not made it to the meeting, I want to receive audio content of the decisions taken."*

A. Information Retrieval with Natural Language Analysis

1) *Tokenizer:* the result of this procedure is as follows:

"*I –Want –to- receive – traffic – reports – of – Bogotá…*"

2) *Filter Words:* removes unimportant words for the request (in this case, words as: I - want are removed), resulting*:*

"To r*eceive – traffic – reports – Bogotá…*"

3) *Words Tagging:* labels and classifies the words obtained before as follows*:*

"To *receive: Verb – traffic: Noun...*"

4) *Named Entity Recognition:* the system classifies the words into "functional" or "control" categories, grouping the words of the functional category in blocks separated by words of the control category, resulting in:

"Block 1: Receive (functional) –traffic(functional) – reports(functional)… Control: before (control) – and(control) – if(control)…Block 2… "

5) *Semantic Analyzer:* at this point we detail the semantic disambiguation of the words classified as functional, based on dictionary [11]. For this example, "report" can be verb or noun, and may have several meanings including: *a written document describing the findings of some individual or group*, *a short account of the news*, and others, from which the system determines the second choice as relevant to this case, using a variant of the *Lesk algorithm* mentioned in [9].

B. Recommender

1) *Recommender Services:* it compares obtained keywords classified as functional with a flow ranking recommendation repository, considering the case for the existence of some match with the words: *"traffic reports"*, the result shown to the executive, is a list of generic flows with services, outcome from previous requests to the system:

"First: SendSMS to GetTraffic …
Second: GetPosition to GetTraffic to SendSMS …"

The executive doesn't select any recommendation, so the remaining processing continues.

2) *Non Functional Requirements Recommender:* this recommender searches non functional parameters from the *Non Functional Repository* using the obtained keywords classified as functional, the result, considering the case for the existence of matches with the words: *"traffic reports"* is as follows:

"*Precision, real-time ...*"

The executive chooses the option *"precision"* and it's accepted by the system.

C. Semantic comparison between the processed request and Service Cluster

1) *Matcher functional parameters:* the input for this stage is represented by two blocks: *" receive traffic reports Bogotá message minute meeting"* and *"receive content audio decision"*, now assuming that, from the process of comparison and service retrieval, the following services were obtained: for the first block: *SendSMS, GetTraffic, GetSMS, SetUpMeeting, AlertMeeting, GetMMS*, for the second block: *GetAudioContent, SendAudioContent, SendMMS, getMMS, GetPosition*. These services are organized according to the functional parameters

comparison (goals, inputs and outputs) getting two ranking of services for each input block respectively: *1-AlertMeeting, 2-GetTraffic,3-GetSMS, 4-GetMMS, 5-SetUpMeeting and 6-SendSMS; and 1-GetPosition, 2-GetMMS, 3-GetAudioContent, 4-SendAudioContent, 5-SendMMS*.

2) Ranking Generator: The system waits a determined time for the input of non-functional parameters, as the executive chose the parameter "*precision*", the retrieved ranking of service is subjected to a weighting based on this non-functional parameter, generating a new ranking for the two blocks: *1-AlertMeeting, 2-GetTraffic, 3-GetMMS, 4-GetSMS, 5-SendSMS and 6-SetUpMeeting; and 1-GetPosition 2-GetAudioContent, 3-GetMMS, 4-SendAudioContent and 5- SendMMS.*

D. Retrieval-based inference

1) Generation of the generic flow control: as a result we have two generic blocks and the words classified as control. *"before and if not"*.

2) Association of ranking and flow: in this phase, the system replaces the generic blocks generated in the previous phase, for the list of services retrieved for each block.

3) Result Storage: the words are stored keeping a relationship with the services that were retrieved, given the possibility of future requests, reduces processing time.

V. CONCLUSION

In this article, we presented an approach that speeds up creation of services in converged environments, critical issues in service deployment by companies in the telecommunications sector. The proposed architecture starts off from a request made by the user in NL from a mobile device and delivers a generic control flow as output which identifies the most suitable services to users. Our proposal also includes the use of top level ontology, which provides greater performance at services selection and also uses a requests-services repository, where records that speed up retrieval time are stored. No incoming requests restriction allows inexperienced users to make requests to the system, unlike other solutions that use templates restricting user's expression.

As a complementary work, we can consider methods for making non-textual requests, like voice, adding a linguistic level (phonetic) to the NLP, increasing the range of end-users. Also, we do not discard the possibility of considering user information, such as Profile and Context.

REFERENCES

[1] Pop F.-C., Cremene M., Tigli J.-Y., Lavirotte S., Riveill M., and Vaida M., Natural Language based On-demand Service Composition. International Journal of Computers, Communications & Control, 2010. V (4): pp. 871-883.

[2] Bandara, A., Semantic Description and Matching of Services for Pervasive Environments, in Engineering, Science and Mathematics 2008, Universidad de Southampton Southampton. pp. 100-150.

[3] ITU-T, General overview of NGN, in Scope and Propose 2004, ITU-T. pp. 10.

[4] Corrales, J.C., Behavioral matchmaking for service retrieval, in Computer Science 2008, University of Versailles Saint-Quentin-en-Yvelines: Versailles. pp. 88.

[5] Sutthikulphanich, K., A Demonstration on Service Compositions based on Natural Language Request and User Contexts, in Telematics 2008, Norwegian University of Science and Technology: Trondheim. pp. 172.

[6] Al-Masri, E. and Q. Mahmoud, Discovering the Best Web Service:A Neural Network-based Solution, in Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics 2009: San Antonio, TX, USA. pp. 4250-4255.

[7] IBM (2011). "What is Watson?" Retrieved August 13, 2011, from http://www-03.ibm.com/innovation/us/watson/what-is-watson/index.html.

[8] Bosca, A., Corno F., Valetto G., and Maglione G., On the fly Construction of Web Services Compositions from Natural Language Requests. Journal of Software, 2006. 1(1): pp. 40-50.

[9] Tarkoma, S., Prehofer, C., Zhdanova A., Moessner K., and Kovacs E., (2007). SPICE: Evolving IMS to Next Generation Service Platforms. International Symposium on Applications and the Internet Workshops. Hiroshima, Japan: pp. 6.

[10] Cordier, C., and Kranenburg, H., Specification of the Knowledge Management Framework, The SPICE project, January 2006, pp 1-49.

[11] Nica, I., Automatic semantic disambiguation, in Language and communication 2002, University of Barcelona: Barcelona.

[12] GateNLP (2011). "Developing Language Processing Components with GATE Version 6" Retrieved August 15, 2011 from http://gate.ac.uk/sale/tao/split.html.