Dynamic Reconfiguration of Composite Convergent Services Supported by Multimodal Search

Armando Ordóñez[1], Hugo Ordóñez[2,3(✉)], Cristhian Figueroa[3,4], Carlos Cobos[5], and Juan Carlos Corrales[3]

[1] Intelligent Management Systems, Fundación Universitaria de Popayán, Popayán, Colombia
armando.ordonez@docente.fup.edu.co
[2] Research Laboratory for Software Engineering, Universidad de San Buenaventura, Cali, Colombia
haordonez@usbcali.edu.co
[3] Telematics Engineering Group, Universidad Del Cauca, Popayán, Colombia
jcorral@unicauca.edu.co
[4] Software Engineering Group, Politecnico di Torino, Turin, Italy
cristhian.figueroa@polito.it
[5] Information Technology Research and Development Group, Universidad Del Cauca, Popayán, Colombia
ccobos@unicauca.edu.co

Abstract. Composite convergent services integrate a set of functionalities from Web and Telecommunication domains. Due to the big amount of available functionalities, automation of composition process is required in many fields. However, automated composition is not feasible in practice if reconfiguration mechanisms are not considered. This paper presents a novel approach for dynamic reconfiguration of convergent services that replaces malfunctioning regions of composite convergent services considering user preferences. In order to replace the regions of services, a multimodal search is performed. Our contributions are: a model for representing composite convergent services and a region-based algorithm for reconfiguring services supported by multimodal search.

Keywords: Convergent services · Dynamic reconfiguration · Multimodal search

1 Introduction

A composite convergent service (CCS) may be defined as a structured set of services (telecommunication and Web services) that works in a coordinated manner to achieve a common goal [1]. CCS achieve the integration and composition of services offered by IT providers with Telecom operators towards the Web Telecom convergence [2].

W. Abramowicz (Ed.): BIS 2015, LNBIP 208, pp. 127–139, 2015.
DOI: 10.1007/978-3-319-19027-3_11

One example of convergent process is a service that manages environmental early warnings. Environmental manager is in charge of decision making about environmental alarms and crops. In order to do so, it is required information from sensor networks, Telecommunication and Web services that process data and that send information to farmers. One typical requirement of such systems is: to emit an alarm to every farmer within a radius of 2 miles from the river if the river flow is greater than 15 % of average. For solving this request, the sensor data are evaluated and if necessary, an emergency map is generated. This map is created drawing a radius of 2 miles from the sensor. To do so, the system may use geographic services and maps from internet. Finally, the system informs about the alarm to farmers inside the emergency area, in this case the best way to send the information is selected (SMS or call). In both cases, services from Web and Telecommunications work in coordination.

Nowadays, composition of convergent services has been increasingly adopted in telecommunication companies in order to create and offer new CCS that integrate different functionalities of existing convergent services. It helps them to avoid developing new CCS from the scratch and duplication of functionalities. However, composing convergent services is not trivial and is time consuming due to the complexity of the integration of different and heterogeneous messages, operations, and data-types in these services [3]. Due to the latter the automation of this composition has been studied actively. Furthermore, the demanding requirements of the telecommunications environment regarding to performance, availability and accuracy, make it necessary that composed convergent services can replace services (nodes) that may be unavailable or malfunctioning during execution and that can impact negatively in the user experience [4].

Malfunctions or unavailability of convergent services may be originated from diverse factors such as network failures or server overload. In such scenarios, CCS must be fixed or reconfigured in order to continue providing their functionalities. Accordingly, reconfiguration and dynamic composition have been identified as leading challenges in Service Oriented Architectures [5].

Previous works in the literature have addressed the reconfiguration of CCS based mostly in replacement of individual failing services [6]. However, existing proposals leave aside Telecom considerations such as service deployment, real-time requirements or event-based interactions [5]. In addition, replacement of failing services for other ones with different non-functional features cannot maintain initial user constraints [7].

This work proposes a dynamic reconfiguration approach supported by a multimodal model for searching services or sets of services that can be used to reconfigure a CCS where there is a set of services (regions of services) that are malfunctioning or unavailable. The multimodal search is a branch of the information retrieval (IR) that aims for efficiently discovering different kinds of content [8].

The region-based reconfiguration is inspired by the work of Lin et al. [5] which describes a mechanism for selecting a replacement for a region surrounding the failing services instead of replacing the whole CCS. A failing region may

be defined as a subset of services from the CCS that needs to be replaced in order to continue the execution of the service without recreating all the CCS. Previously the region-based reconfiguration mechanism was tested in the AUTO platform using automated planning [9–11], in this paper we focus on the multimodal search instead of the automated planning. Multimodal search uses textual and structural information in order to cover more information dimensions for querying during the search [12], this makes that the retrieval of services to be more precise than traditional information retrieval techniques.

The rest of this paper is organized as follows. Section 2 depicts a general overview of AUTO. Section 3 presents the modeling of CCS. Section 4 shows the service adaptation and the multimodal search applied to reconfiguration process. Section 5 presents the experimentation. Finally, Sect. 6 includes the conclusions as well as the relevant related work.

2 The Approach for Dynamic Reconfiguration of Composite Convergent Services

Figure 1, shows the architecture the main modules of AUTO [11]. AUTO, defines sequential phases for service composition namely: creation, synthesis and execution.

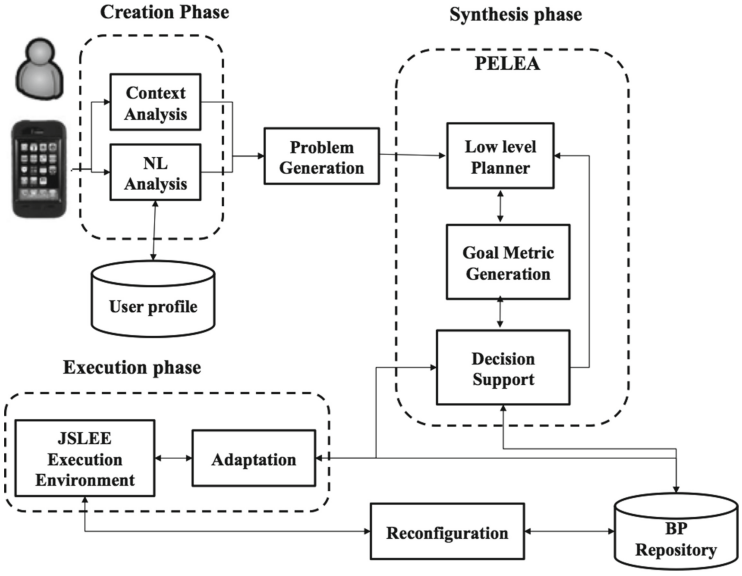

Fig. 1. Architecture of the approach for dynamic convergent service composition

The creation phase accepts user queries (requests) expressed either as voice or as text strings in natural language (NL). The NL Analysis module deals with each query in order to extract a set of significant words, i.e. to delete "stop words", reduce words to their lexical root etc. The Context Analysis module

enriches the user queries with information about the user preferences, device capabilities and situational context. The Problem Generation module translates the processed user query into a problem file expressed in the planning language PDDL (Planning Domain Definition Language) [13]. Thus, this problem file contains the goals (user objectives) and metrics (preferences of the user about the composition). Additionally, it can also contain information about the initial state such as location, device information, etc.

The synthesis phase, receives the problem file as input and generates the structure of the composite convergent service CCS. To do so, this phase uses automated planning. In automated planning the generated structures are known as plans. A plan contains a set of services that solves user queries, the obtained plan represents the CCS. This phase is framed in the Planning and Learning Architecture (PELEA) [14]. *PELEA* is an architecture that contains modules for plans generation, monitoring and reconfiguration. Specifically, the low level planner performs the plan generation, the Goal Metric Generator module associates the goals and metrics, obtained from the input problem, with services available for the composition. Finally the decision support performs the monitoring and reconfiguration of the process. The synthesis phase requires that the user receives the plan immediately, therefore the elapsed time spent building the plans must be minimized. Nevertheless, in order to get the plan that represents the best solution, a big computational effort is necessary. On the average case finding the optimal solution requires exploring a very significant part of the search space, which makes such an endeavor impractical. Therefore, the best solution given a time window may be acceptable to begin the execution, and afterwards the planner may refine the plan while the first services are invoked and the associated tasks are performed in the real world. These new plans are potential replacements for the initial CCS, so these CCS are stored in the *BP repository*, which also contains simple convergent services (i.e. those that are not composed with others into a CCS).

The execution phase is mainly performed in the execution environment for telecommunication applications named Java Service Logic Execution Environment (JSLEE). The Adaptation module integrates *PELEA* and JSLEE in order to take the CCS (synthesized plans) and create executable CCS. To do so, the adaptation module associates the planning operators to Java Snippets and generates a CCS using JSLEE Service Building Blocks (SBB). SBBs are the basic components of the JSLEE architecture and are able to call external Web services or Telecom functionalities AUTO monitors CCS execution and repairs plans.

3 Modeling of Services for Multimodal Search

Our approach integrates multimodal search principles into CCS similarity detection during reconfiguration. Multimodal search is a branch of information retrieval (IR) that allows to efficiently discovering different kind of content - such as text, images, and video. Thus, the multimodal model for searching CCS unifies in single search space textual information and the structure of the CCS.

In this case, linguistics represents names and descriptions of the CCS elements (e.g. activities, interfaces, messages, gates, and events), and the structure is defined as codebooks formed of n-structural components ruled by the sequentially of the CCS control-flow (i.e. the union of two or more control-flow components. It has the following advantages: greater speed and precision, diversity in query types, and a good representation of CCS that allows delivery of results that correspond more precisely to user requirements [12].

Our reconfiguration algorithm that is based on the multimodal approach for searching service regions, a CCS must be appropriately represented so it can be dynamically composed and reconfigured. Therefore we have defined a model for representing a CCS as text strings considering users preferences, context conditions and devices capabilities. We have selected some dimensions associated with the devices and the information that can be obtained about the situation of the users using the analysis of their requests.

These dimensions are based on the context analysis module (Fig. 1). It classifies the information according to three dimensions: Device, User Profile and Situation (See Fig. 2). Each one of them obtains the information from different sources and defines the selection of domains. These dimensions allow us to analyze the capability of the system for responding to user preferences.

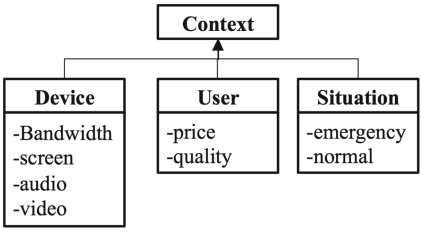

Fig. 2. Dimensions of user Context: Device, User and Situation

The Device dimension gathers the information from the device and the network. To access this information, the device ID or model is consulted in capabilities repositories like the Wireless Universal Resource File (WURFL) and the Composite Capability/Preference Profiles (CC/PP). These repositories store technical specifications of different commercial communication devices.

The User Profile dimension gathers the information using the user ID to look up his/her preferences in the preferences repository in the system. The Situation dimension analyses the Natural Language request identifying specific words such as “urgently” or “emergency”. The relationship between these dimensions is established using a preferences function, which assigns weights to each one of them. For instance, considering a user connected that employs a smartphone with video capabilities. This user has registered low cost preference; therefore, cheaper services like SMS or regular voice calls are more likely to be selected because the user price preference has a higher weight in the preferences calculation. On the other hand, if the system detects a situation of emergency, the user preferences from the repository are overridden and the most reliable services are selected instead no matter the cost of the service.

Table 1. Mapping between user context information and service properties

User criteria	Service property	Type
Network	Payload size	Bytes
Device	Payload size	Bytes
Location	Voice, text	Integer
Data subscription	Require subscription	Boolean
Only free services	Cost	Value
Voice subscription	Voice, text	Boolean
Delivery quality	Delivery warranty	Integer

To map the relationship between context and services, the Context Analyzer maps context data to services properties. This information is used in subsequent stages of Service composition, as described further on. Table 1 shows context criteria identified from the request. It additionally, shows how these user criteria are mapped to service properties. These properties are later considered during CCS reconfiguration.

4 Service Adaptation

4.1 Automatic Deployment

As mentioned before, in automated composition it is required that the user to be served immediately, so the best solution given a time window is selected whereas additional plans generated after the initial solution comprise the ranking of alternative plans that are possibly used if required (see Fig. 3). This approach is described in [11]. During the synthesis of the input problem, the best plan that satisfies the user request and preferences in a defined time is selected. The computation and execution of alternative plans is done in background so the user gets an instant response. In order to establish an association between dynamic synthesized plans and JSLEE components, the synthesized CCS are translated into Java components. To do so, the abstract CCS are integrated with execution patterns (conditional, fork, join, ...) defined as Java snippets [11].

The reconfiguration process uses an alarm-based approach. These alarms are described in the JSLEE standard. In case of failure, one or more alarms are activated. The alarms are encrusted in the Java code during the translation between abstract CCS into SBB Components. The black circles in the Fig. 3 represent such alarms.

The alarms encrusted in the executable CCS allow monitoring the execution during the workflow. Monitoring the CCS may lead to three courses of action: if the error is caused by an atomic service, the system selects another service from the implementation services repository and continues on; second, if the problem is caused by the whole CCS and the CCS itself can be modified, a new plan from

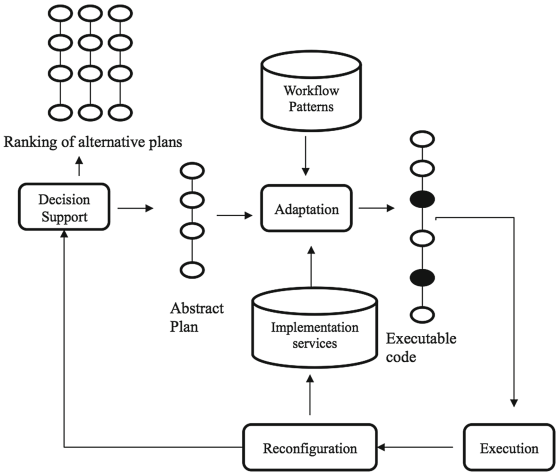

Fig. 3. Reconfiguration schema

the previously generated ranking is selected; finally, if there is a problem in the middle of the executing CCS and no alternatives are available, a new planning CCS is initiated in the Decision Support module to complete the task, starting from the actual state of the world, i.e., a set of values of the variables that define a system in a given moment.

To solve these issues, our approach incorporates a region-based reconfiguration. Next we present the region-based service reconfiguration algorithm.

In the traditional reconfiguration, if the failure is raised in more than one service, it could be necessary to re-plan and/or re-adapt the entire CCS. Besides, it could be necessary to undo all the performed tasks in the previous plan that does not match with the new generated plan. Consequently, the less reconfiguration is performed; the better would be the performance [12]. It was important to clarify that undo actions aren't feasible in some situations when services have been invoked. However, in some scenarios such as early warning management, the services or actions are to configure the sensing time of some devices, or turn on other device. In the latter situations the undo actions may be performed. Additionally, in some scenarios, the reconfiguration of telecommunication services may be done in execution time using techniques such as code injection [15]

In this paper we model the CCS as text strings and perform a novel search algorithm for discovering regions of services based on the multimodal approach. Therefore, it is important to find CCS or fragments thereof that can be reusable for defining new adjustable CCS to meet different requirements of the initial composition. One of the current challenges in this field is precisely the improvement in efficiency in the search for such CCS or fragments that normally is done manually or automated planning, involving heavy demands on time and resources.

4.2 Reconfiguration Algorithm

The Algorithm 1 for convergent service reconfiguration is called when an error is detected during execution, the algorithm is shown below:

Algorithm 1. Reconfiguration Algorithm

Require: faulty CCS si, threshold c
Ensure: replaced sub process $Rf = ri$
1: Set region $ri = si$; $Rf = \emptyset$, $counter = 0$
2: **while** $Rf \neq \emptyset$ and $counter < threshold \quad c$ **do**
3: $counter++$
4: $ri = \{ri - counter, ri, ri + counter\}$
5: Select a created plan ri' that meets QoS of ri
6: **if** $ri' \neq \emptyset$ **then**
7: add ri' to Rf
8: **end if**
9: **if** $Rf = \emptyset$ **then**
10: Search a Rf similar to si using multimodal search
11: **if** $Rf \neq \emptyset$ **then**
12: add ri' to Rf
13: **end if**
14: **end if**
15: **end while**
16: **return** Rf

Let us suppose that a plan is generated and a CCS si presents malfunctions. Then, a replacement for si in the repository must be found. The new CCS should contain the same non-functional properties as to the initial constraints of the plan defined by the user's preferences.

If such a CCS cannot be found, it is necessary to try to replace the surrounding services, so we increase the counter and expand the region. To do so, we include the previous and the next service. Replacing a set of services together gives us more flexibility as long as the replacing services can meet the combined constraints needed. In this way, the reconfiguration region can be extended or reduced in order to include services that accomplish with the aforementioned constraints. It is worth noting that if a region includes many services for replacement, selecting alternative plans from the ranking is probably a better option.

Given p faulty services and a reconfiguration threshold c ($0 < c < 1$) on the maximum number of services to be repaired, the algorithm first starts a loop (lines 2–15) to repeatedly expand the region. Inside the loop, the algorithm first tries to find a replacement region ri' for the failing service and adds it to the response Rf (lines 5–8). If the required plan does not exist, the algorithm search for similar CCS stored in the repository in order to recompose the region ri. The CCS are found using a multimodal model search approach (lines 9–13), which receives as input a text string representing the faulty services to be replaced.

4.3 Multimodal Search for Convergent Services

The multimodal search of CCS is intended to search for text string representing a convergent service or a fragment thereof. This model transforms the CCS into a matrix for two components: a linguistic component and a structural component. The linguistic component contains all the convergent services and gates that compose the CCS, and the structural component contains pairs of sequential nodes (service-service or service-gate) maintaining the order they appear in the CCS. Next the formation of the linguistic and the structural components is detailed.

Formation of the Linguistic Component: suppose that we have a repository T of CCS: $T = \{CCS_1, CCS_2, \ldots, CCS_i, \ldots, CCS_l\}$, where l is the total number of CCSs that it contains. The first step of the algorithm is to read each CCS_i and to represent it as a tree A. Then the algorithm takes each A_i, extracts the textual characteristics (activity name, activity type, and description) and forms a vector $Vtc_i = \{tc_{i,1}, tc_{i,1}, \ldots, tc_{i,l}, \ldots, tc_{i,L}\}$, where L is the number of textual characteristics $tc_{i,l}$ found in A_i. For each vector Vtc_i, which represents a CCS_i, a row of a matrix of textual components MC is constructed. This row contains the linguistic component for the CCS stored in the repository. In this matrix, i represents each CCS and l a textual characteristic for each of them.

Formation of the Structural Component: a codebook Cd is a set of N structural components describing one or more nodes of the CCS in the form of text strings. The set of codebooks formed from the whole repository is called the codebook component matrix. This matrix is formed by taking each tree A_i, which all contain a vector of codebooks $Vcb_i = \sum_{k=1}^{p} Cd_{i,k}$. Therefore, the codebook component matrix MCd_{ik} is formed where i represents the current CCS and k represents its correspondent codebook.

Indexing Process: the linguistic and codebook components are weighted to create a multimodal search index MI composed of the matrix of the linguistic component MC and the codebook component matrix MCd, i.e. $MI = \{MCd \cup MC\}$. The index also saves the reference to the physical file of each of the models stored in the repository. These models correspond to CCS stored in the BP Repository.

– Weighting: Next, the indexing layer applies a weighting scheme of terms, similar to that proposed in information retrieval (IR), via the document representation vector model to form the term document matrix. This weighting scheme is based on Eq. 1, initially proposed by Salton [16].

$$W_{i,j} = \frac{F_{i,j}}{max(F_i)} \times \log\left(\frac{I}{I_j + 1}\right) \tag{1}$$

In Eq. 1, $F_{i,j}$ is the observed frequency of a component j in CCS_i, the component j may be a linguistic (l) or codebook component k. $max(F_i)$ is the greatest frequency observed in CCS_i, I is the number of CCSs in the collection. Each cell $W_{i,j}$ of the multimodal index matrix reflects the weight of a specific element j of a CCS_i, compared with all the elements of the BP in the repository.

– Index: The multimodal index is stored in a physical file within the file system of the operating system, in which each CCS is indexed through a pre-processing mechanism. This mechanism consists in converting all the terms of the linguistic and codebook component matrices to lowercase and removing stop words, accents and special characters. Subsequently, the stemming technique (Porter algorithm [16]) is applied, which enables each of the matrix elements to be transformed into their lexical root (e.g. the terms "fishing" and "fished" become the root, "fish") and physical file of each CCS to be stored in the repository.

Search Similar CCS: when a CCS fails and there are not other services or CCS that accomplish their IOPE and QoS requirements, the multimodal search process is called. It receives the malfunctioning CCS and forms the linguistic and the structural component. Next, a conceptual rating (score) is expressed expressed by Eq. 2 (based on the *Lucene practical scoring function*[1]) is applied in order to return a ranking list of results containing a set of convergent services with the highest conceptual punctuation within the multimodal index.

$$score\,(q,d) = coord\,(q,d) \times \sum_{t \in q} \left(tf\,(t \in d) + idf\,(t)^2 \times norm\,(t,d) \right) \qquad (2)$$

In Eq. 2 t is a term of query q; d is the current CCS with q; $tf(t \in d)$ is the term frequency defined as the number of times the term t appears in d so that CCS are ranked according to the values of term frequency scores; $idf(t)$ is the number of convergent services in which term t appears (inverse frequency); $coord(q,d)$ is the scoring factor based on the number of query terms found in the queried CCS (those CCS containing the most query terms are scored highest); $norm(t,d)$ is the weighting factor in the indexing, taken from $w_{i,j}$ in the multimodal index

Eventually, the goal of the planning algorithm is to create a new plan. Furthermore, the planning algorithm considers user preferences trough the cost function that assigns a cost value to each generated plan. This process is described in detail elsewhere [10].

The algorithm continues and, if the repaired regions include too many nodes (as defined by c), the whole plan will be replaced from the existing ranking of plans.

[1] http://lucene.apache.org/core/3_6_2/api/core/org/apache/lucene/search/Similarity.html.

5 Experimentation

The experimentation was focused on computing the performance of the proposed approach which is based on the execution time evaluated in two parts: the first part verifies the performance of the proposed approach in a telecommunications environment and the second part verifies the performance of the multimodal search using test plans with different number of nodes.

For first part, an hypothetical composite convergent service was defined by running in it a Telecom Server that invokes some telecom features: send SMS, invoke a Web Service and performs a voice call. The composite service uses a set of services in a sequence. The traditional method reconfigures the whole plan from scratch, to do so; this method must undo the tasks performed before to the error occurs. Only then the new plan can be started. If the error is presented at the beginning, then there is no need for undo any task. Conversely, if the error is present at the last node it is necessary to remake the entire plan or select other plan from the ranking. For representing undo tasks in our experiment, we performed a call establishments and web service invocations. As would be expected, time increases linearly with the number of tasks that must be undone; the higher the node of the error, the higher is the time consumed to undone the previous tasks (continuous line in Fig. 4).

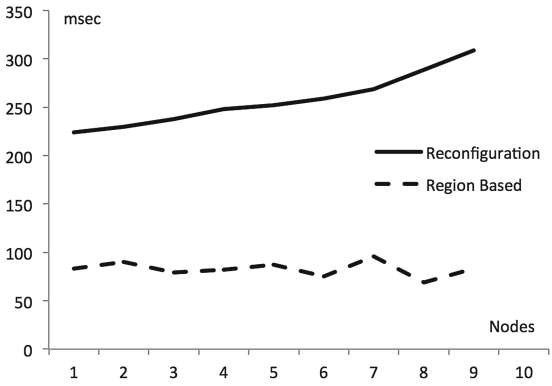

Fig. 4. Performance comparison of traditional vs. Region based reconfiguration using automated planning

Figure 4 shows the performance of the reconfiguration algorithm using automated planning. Figure 4 shows a test a plan with 10 nodes. It is assumed than an error occurred at different nodes of the path from 1st to 10th node (X axis in Fig. 4).

As can be seen in Fig. 4, the algorithm for region-based reconfiguration using automated planning has a better performance that the traditional reconfiguration processes. Next, the aim of the experimentation is to test if the multimodal search presents better results that the region based reconfiguration using automated planning.

Table 2. Performance of the multimodal search for CCS with 10, 20 and 30 nodes

10 Nodes runtime (ms)	20 Nodes runtime (ms)	30 Nodes runtime (ms)
17	40	60

For the second part, the performance of the multimodal search was evaluated finding CCS with 10, 20, and 30 nodes taking into account that most of the CCS contain only few nodes. Table 2 shows, that the Multimodal search approach performing a simple text extraction algorithm (complexity $O(n2)$) takes just a few milliseconds to retrieve the results (17–60 ms) in the CCS with 10–30 nodes. Therefore the multimodal algorithm is suitable to the process of reconfiguration of CCS as it shows a low computational complexity.

6 Conclusions

Convergent composition requires that CCS can be efficiently recovered from failures. This work presents the results of our ongoing work towards the definition of a mechanism for planning based reconfiguration in convergent domains. Furthermore, we present an algorithm based on multimodal model to find services that can be used to reconfigure troublesome regions of a CCS instead of remaking the whole CCS or selecting other plan. "Replanning" the whole CCS or selecting other plan from the ranking involves a big effort in undoing the actions of services previous to the failing service.

Multimodal search is based on text and structural information; due to the latter the quality of the CCS retrieval is higher than using automated planning. Besides, the inclusion of structural information helps to get better plans that sequential plans obtained from traditional planners. Finally the experimental results show that the performance of the multimodal search may reduce in many orders of magnitude the time of execution.

The future work will be focused in using the algorithm for performing the reconfiguration of different failing regions at the same time. Equally we are interested in perform further testing that evaluates performance and quality of the approach in Cloud based platforms for convergent services measuring real user experience.

References

1. Object Management Group: Uml profile for advanced and integrated telecommunication services (TelcoML). Standard, OMG, August 2013. http://www.omg.org/spec/TelcoML/1.0/
2. Ambra, T.: Description and composition of services towards the web-telecom convergence. In: Lomuscio, A.R., Nepal, S., Patrizi, F., Benatallah, B., Brandić, I. (eds.) ICSOC 2013. LNCS, vol. 8377, pp. 578–584. Springer, Heidelberg (2014)

3. Wang, D., Yang, Y., Mi, Z.: A genetic-based approach to web service composition in geo-distributed cloud environment q. Comput. Electr. Eng. (2014)
4. Jula, A., Sundararajan, E., Othman, Z.: Cloud computing service composition: a systematic literature review. Expert Syst. Appl. **41**(8), 3809–3824 (2014)
5. Lin, K.J., Zhang, J., Zhai, Y., Xu, B.: The design and implementation of service process reconfiguration with end-to-end QOS constraints in SOA. Serv. Oriented Comput. Appl. **4**(3), 157–168 (2010)
6. Pernici, D.A.B.: Adaptive service composition in flexible processes. IEEE Trans. Softw. Eng. **33**, 369–384 (2007)
7. Kaldeli, E., Lazovik, A., Aiello, M.: Continual planning with sensing for web service composition. In: AAAI, pp. 1198–1203 (2011)
8. Ordóñez, H., Corrales, J.C., Cobos, C.: Multisearchbp-entorno para busqueda y agrupacion de modelos de procesos de negocio. Polibits **49**, 29–38 (2014)
9. Ordóñez, A., Corrales, J.C., Falcarin, P.: Natural language processing based services composition for environmental management. In: 2012 7th International Conference on System of Systems Engineering (SoSE), pp. 497–502. IEEE (2012)
10. Ordonez, A., Alcázar, V., Borrajo, D., Falcarin, P., Corrales, J.C.: An automated user-centered planning framework for decision support in environmental early warnings. In: Pavón, J., Duque-Méndez, N.D., Fuentes-Fernández, R. (eds.) IBERAMIA 2012. LNCS, vol. 7637, pp. 591–600. Springer, Heidelberg (2012)
11. Ordóñez, A., Alcázar, V., Corrales, J.C., Falcarin, P.: Automated context aware composition of advanced telecom services for environmental early warnings. Expert Syst. Appl. **41**(13), 5907–5916 (2014)
12. Ordoñez, H., Corrales, J.C., Cobos, C.: Business processes retrieval based on multimodal search and lingo clustering algorithm. IEEE Latin Am. Trans. **13**(9), 40–48 (2015)
13. Gerevini, A.E., Haslum, P., Long, D., Saetti, A., Dimopoulos, Y.: Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. Artif. Intell. **173**(56), 619–668 (2009). Advances in Automated Plan Generation
14. Guzmán, C., Alcázar, V., Prior, D., Onaindia, E., Borrajo, D., Fdez-Olivares, J., Quintero, E.: PELEA: a domain-independent architecture for planning, execution and learning. In: Proceedings of the ICAPS, vol. 12, pp. 38–45 (2012)
15. Adrada, D., Salazar, E., Rojas, J., Corrales, J.C.: Automatic code instrumentation for converged service monitoring and fault detection. In: 2014 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 708–713. IEEE (2014)
16. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval, vol. 1. Cambridge University Press, Cambridge (2008)