Expert Systems with Applications 41 (2014) 5907–5916

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Automated context aware composition of Advanced Telecom Services for environmental early warnings

Armando Ordonez [a,*], Vidal Alcázar [b], Juan Carlos Corrales [a], Paolo Falcarin [c]

[a] University of Cauca, Cll. 5 4-70 Popayán, Colombia
[b] Universidad Carlos III de Madrid, Av. Universidad 30, 28911 Leganés, Spain
[c] University of East London, Docklands Campus, London E16 2RD, United Kingdom

ARTICLE INFO

Keywords:
Automated planning
Advance Telecom Services
Services composition

ABSTRACT

This paper presents one of the main components of a framework for automated composition of Advanced Telecom Services for environmental early Warnings. The framework, called AUTO, is composed by three main modules: a request processing module that transforms natural language and context information into a planning instance; the automated planning module, based on PELEA, an architecture for planning and execution; and the Service Execution Environment Advance Telecom Services. This paper focuses on the description of the translation of the user request in natural language and his context into planning instances. These planning instances represent service composition tasks based on Automated Planning. The advantages of this approach, like the automatic inclusion of context and user preferences in the composition of services, will be presented. Also, the current implementation will be described and some experimentation will prove the viability of AUTO.

1. Introduction

An Advanced Telecom Service is defined as a service that exploits the convergence of communication networks and takes also advantage of features accessible from the World Wide Web (Object Management Group [OMG], 2012). This kind of composite services are needed when the request of a client cannot be satisfied by a single service. Traditionally, the composition of Telecom Services in the industry is carried out through interactive graphical interfaces called Services Creation Environments (SCE),[1] which allow the user to articulate services functionalities using drag-and-drop tools. This approach is usually valid in Telecommunications thanks to the fact that the composition is performed using exclusively Telecom services, whose reliability is very high (99.999%) (Chrighton, Long, & Page, 2007). However, composing Advanced Telecom Services including services from the Internet, commonly called Web Services, is a very different matter. Due to the dynamic nature of the Internet, Web Services may change, become unavailable and grow in number to unmanageable sizes. This means that composition of advance telecom services is unfeasible in practice if the user employs traditional methods.

Previous works from both the academia (Hatzi et al., 2012; Hoffmann, Bertoli, & Pistore, 2007) and industry[2] have revolved around this topic. However, few academic approaches include implementations in real world scenarios, and few works from the industry are publicly open and easy to extrapolate to similar cases, as exposed by Dustdar and Schreiner (2005) Furthermore, previous work in the area focuses on representing the services using semantic descriptors (Huang, Lee, & Crespi, 2012) and applying afterwards different techniques, such as automated planning (Oh, Lee, & Kumara, 2006) or genetic algorithms (Ye, Zhou, & Bouguettaya, 2011). Additionally, some of these approaches have obtained promising results using Web Services, but no prior work that we are aware of is able to combine them with Telecom features. Another important limitation of these approaches is that they require that the users express their requests using formal languages such as Golog (Sohrabi, Prokoshyna, & McIlraith, 2009) or PDDL (Hatzi et al., 2012). This means that the interaction of a large number of people with little or no technical background with the system is unfeasible, which renders these works useful only in a limited set of scenarios.

* Corresponding author. Tel.: +57 2 8209800; fax: +57 2 8209900.
E-mail addresses: jaordonez@unicauca.edu.co (A. Ordonez), valcazar@inf.uc3m.es (V. Alcázar), jcorral@unicauca.edu.co (J.C. Corrales), falcarin@unicauca.edu.co (P. Falcarin).

[1] Some SCEs are Open Cloud Visual Service Architect: http://www.opencloud.com/products/rhino-VSA/ and the IBM Model driven SCE http://www.research.ibm.com/haifa//projects/services/sce/index.shtml.

[2] Some examples from the industry are Zypr http://www.zypr.net/ and SIRI http://www.apple.com/iphone/features/siri.html.

http://dx.doi.org/10.1016/j.eswa.2014.03.045
0957-4174/© 2014 Elsevier Ltd. All rights reserved.

The goal of the environmental early warning field is to create contingency plans that help resolve potentially harmful or dangerous situations, such as floods or tropical storms. The detection of these situations combines both information gathered from sensors around the monitored area and the input of relevant users. An example of such a situation would be evacuating all the villages close to a river after detecting that its level has risen beyond regular measurements or upon the request of an observer. In this case a contingency plan would include actions to monitor the river, determine affected areas, warn the villagers and coordinate the logistics of the evacuation. Since the participation of a human is required in critical scenarios, we assume that all the requests are triggered by users. Besides, in rural environments the access to a device able to send a request in the format specified by the system may be limited. This means that the system should be able to process natural language so it can be initiated through simple communication means like a phone call or an SMS. In fact, thanks to the rather restrictive process of the composition of services in this domain, the main procedures and their associated services can be described using simple semantic annotations by experts in the field, which allow a natural language recognition technique to be implemented without imposing significant restrictions to the potential users. This also means that an automatic translation of the input into a formal language without intervention from either the user or an expert is a much easier task.

Recent studies have explored the application of Natural Language processing techniques to the Automated Composition of Services field, especially in intelligent home environments (Cremene et al., 2009) These approaches offer mechanisms to map user words to basic functionalities of the system. In those works, the goal is to extract the workflow of the composite service from the User Request. The novelty of the application of Natural Language in this work is that in some scenarios, such as the early environmental warnings domain, the user can only express his desire or goal and not the workflow of the composite service. The workflow of the new composite service should be obtained from the automated composition process. Another element of particular relevance is the user preferences and the context of the situation. Automatically discerning the context of the request can enrich the process of automated composition processing by adding important information. In this regard, special words in user requests such as "urgently", "danger" and "quickly" can represent a preference for service quality instead of cost. Secondly, data about the user location and the characteristics of the device may offer critical information about the preferred delivery format for the output of the composition of services.

In relation to the usefulness of automated services composition in this work, some considerations must be made. The Environmental management domain contains the following particularities: (i) composed services can be formed by Web Services and/or by basic Telecom features like Call or send SMS, (ii) the number of different services is rather limited due to the specialized field of action, (iii) in environmental management the procedures for emergencies handling and monitoring are standard. This means that these procedures and the associated services can be described using semantic annotations by experts, as the number of annotations needed is relatively few and standard. This is in fact a very important issue, since few service providers have taken up the opportunity to mark-up their web services, for the simple reason that they do not envisage the use of their services by an automated planning system (Carman, Serafini, & Traverso, 2003). Although the present architecture is focused on a particular domain, that is, the implementation is domain-dependent, most of the principles discussed here are applicable to similar scenarios.

The overall functioning of the architecture is as follows: the user request is received in natural language from a given device. The request is processed to determine the goals and preferences of the user. At the same time, information obtained from the sensors may be added to the request depending on the context. Next, the request is translated dynamically into a planning instance modeled using the Planning Domain Definition Language (PDDL) (Gerevini, Haslum, Long, Saetti, & Dimopoulos, 2009). Then, the PDDL formatted request is sent to the High Level Replanner module of the Planning, Learning and Execution Architecture (PELEA) (Guzmán et al., 2012). PELEA computes a plan using a domain-independent planner and manages its execution (note that the plan represents the composition of services). Finally, the composed plan is executed in a Jain SLEE 5 environment for convergent services.

In this context, the main contributions of our research work are: (1) A Framework for Advanced Telecom Services Composition based on Automated Planning; (2) a metamodel for user context including device characteristics, preferences and user profile information; (3) a mechanism that automatically translates user requests in Natural language into planning instances formulated in PDDL; and (4) the integration of a planning architecture in the execution of plans based on expert-made Mashups.

This paper is organized as follows: Section 2 presents the motivating scenario of early warning in environmental domain. As the AUTO framework is based in Automated Planning; In Section 3 the whole architecture of the framework is described. Section 4 exposes the background about planning and describes the planning domain construction. Section 5 describes the modeling of user request and his translation to PDDL. Section 6 Describes the Prototype and experimentation. Section 7 presents the related work and Section 8 draws the conclusions.

2. Environmental management domain

A sketch of an environmental management system is presented in Fig. 1. The role of the environmental manager is to make decisions about the environmental alarms and crop management of a region. For this purpose he/she can request information from the network of sensors deployed at several spots. The environmental manager can also use Telecommunication and Web services to process basic data and send information to both farmers and actuators. Available services often change dynamically and the resources may be limited.

As the environmental manager comes from fields like biology or agriculture, his/her technological background may be low. This means that the odds of the user knowing formal representation languages are low. Furthermore, electronic devices in the area may be scarce and not reliable or obsolete. Thus, the preferred way to enter information to the system is by voice and in natural language. This way, users do not have to know how the system internally works and can make requests from regular mobile devices or landline phones. Commonly the user expresses his/her request informally; here are two examples:

- "I need to compute the hydrological balance of zone 1 and receive the resulting map in my cell phone".
- "If the river flow of zone 2 is greater than 15% of the safety limit on average, send a warning to every farmer within a radius of 2 miles from the river and create an action map including emergency and rescue groups".

In the first case a composition of services could be: gathering information from the sensors in zone 1, using hydrological services from the Internet to process the sensed data, obtaining the map from Google maps, composing the final image and sending it to the user by MMS. A similar course of action would be done in the second case, making decisions about which maps are available

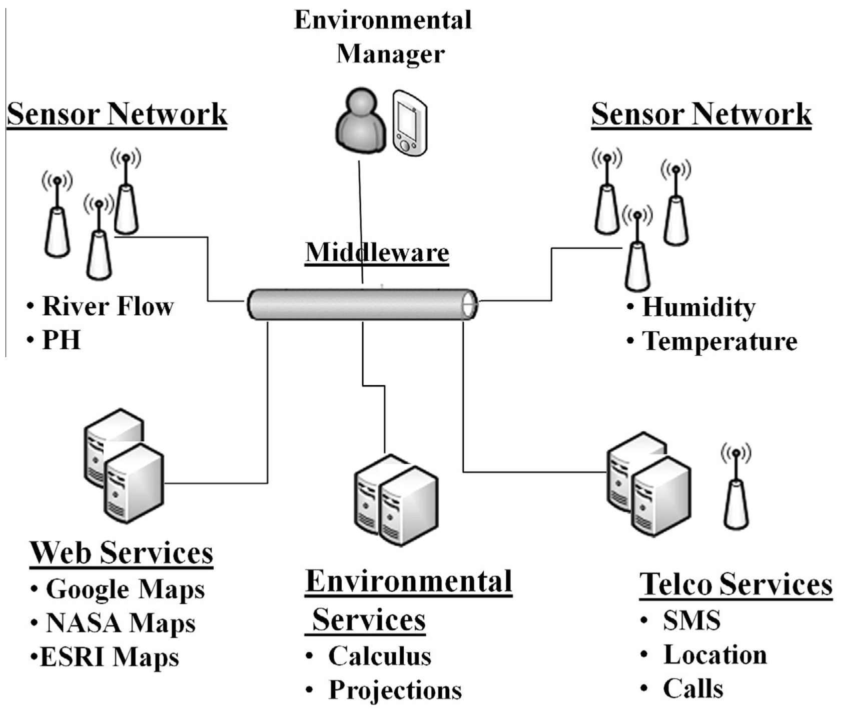

Fig. 1. Elements involved in the Early Warning Domain for Environmental Management.

or best suited to the case, the best way of warning the farmers (i.e. SMS, direct call, e-mail), etc.

In this regard, the next considerations are the foundations of the designing the AUTO Framework.

- Usability for end users: simplicity of use and ubiquity forces the use of natural language recognition techniques. Also all the resulting processes must be readable so they can be supervised and modified by an expert due to the critical nature of the domain.
- Integration with the execution: Automated composition includes integration of Telecom and Web functionalities, which requires specialized platforms.
- Reaction to change: planning systems deployed in dynamic environments must be able to react to potential changes during both the computation of the plans and the execution of the services.
- User preferences: multiple compositions of services are often possible, so the user may express his/her preferences in the request. Typical cases include minimizing the time of the execution of the composed services if it is an urgent request or minimizing the cost if there are no time constraints.

One interesting work in the same sense, has been presented in this sense, WORKAD (Catarci et al., 2011) is a project focused in the development of a two level architecture to support rescue operators during emergency management.

3. Architecture of the AUTO framework

The architecture of AUTO is depicted in Fig. 2. The modules may be deployed in different machines so the computational load of the different processes can be distributed. The access method can be either voice or text, which means that AUTO can be accessed from a broad range of devices. In the literature, other alternatives have been proposed for user request treatment, such as Mashups (Zhao, Bhattarai, Liu, & Crespi, 2011) or service creation environments (Laga, Bertin, Glitho, & Crespi, 2012). However, natural language offers a better mechanism for end users without expertise to express their requests (Lim & Lee, 2010).

The Request Analysis module decomposes the request in constitutive parts: the Context Analyzer and the Natural Language Analysis (NL Analysis). The Context Analysis is based on three elements: user preferences, device capabilities and situational context (see Fig. 3). Additional information may be added to the request from a database containing the profile of the user. The PDDL Generation module makes a translation from the processed request into a problem file in PDDL. The automatically generated problem in PDDL is the input sent to PELEA, which performs the service composition using a domain-independent planner.

The PDDL domain does not change, so PELEA uses the same domain definition for different service composition requests. In situations in which the status and characteristics of the services may change PELEA can monitor the execution and replan or repair the current plan as needed, although in this work we assume that no replannification is needed.

AUTO uses a robust execution environment for telecommunications applications called Java Service Logic Execution Environment (JSLEE). The integration between PELEA and JSLEE is done in the execution module and allows the execution and the sensing of the state of the service composition. The execution module makes a dynamic association between the plan tasks and the JSLEE Service Building Blocks (SBB). SBBs are the basic components of the JSLEE architecture and call external Web services or Telecom functionalities. The association between automated generated plans and SBB is done at run time, so changes in the environment can be easily dealt with.

4. Automated planning and specification of the domain

Automated Planning is the task of finding a sequence of actions (plan) that leads to a state where the goals stated in the definition of the problem are achieved. The goal state is obtained by executing the actions of the plan from a given initial state. A domain

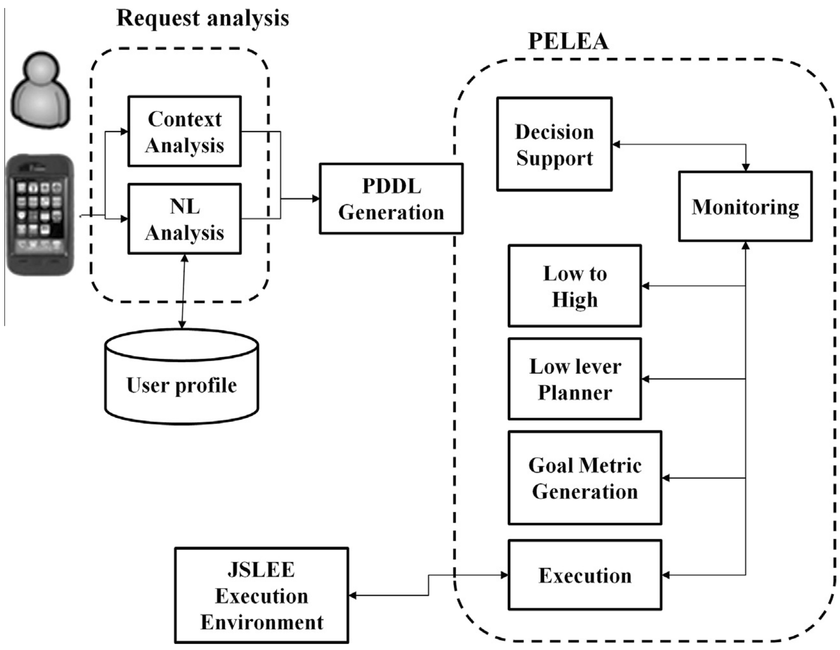

Fig. 2. Architecture of the AUTO framework.

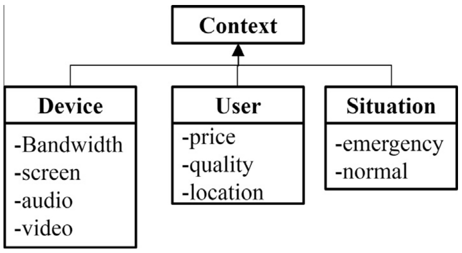

Fig. 3. Dimensions of User Context: Device, User and Situation.

independent planner takes two inputs: a domain description and a problem description. Both are usually described using the standard language PDDL. The planning domain can be understood as a set of statements defining the types of the objects of the domain, its attributes and relationships and the actions that can be performed. In our current case, actions encode services relevant to the final composition of services. On the other hand, the problem file includes the initial state (which may include location, device information, sensor data, etc.); goals (user objectives) and possibly also a metric (user preferences about the service composition process). As long as the domain and problem are specified in PDDL almost all current domain-independent planners can be used to solve the instances of any domain.

Here an overall description of the domain in PDDL will be given. First, the domain includes the definition of the types of the objects that appear in the domain. Objects belong to a single type, but it is possible to define a hierarchy of types that allow a more general definition of the objects. In this case, types represent among others zones, users, communication means, etc.

```
(:types zone user source communication – object
map_step1 map_generation maptype – object)
```

Next, we have to define domain predicates and functions. Predicates are binary variables or propositions that represent whether a defining characteristic of the domain (which may be related to one or more objects) is true. As an example, we define a predicate to express that we have already obtained some coordinates, or that we have already performed some sensing in a given zone. Functions, also known in Automated Planning as fluents, are numeric variables that serve a similar purpose. Predicates and functions have arguments that are typed.

```
(:predicates (coordinates_taken ?z – zone)
   (sensed ?z – zone)
   (isolines ?z – zone) ...)
(:functions (time) (cost)
   (messages ?m – communication)
   (access ?m – maptype)
   (source-cost ?m – source) . . .)
```

Actions in PDDL include typed parameters, preconditions and effects. In our case, the actions of the domain model the different kinds of services. We describe now some of the actions of the domain definition. The *get_coordinates* action has the following parameters: the user that performs the composition, the zone to be analyzed, the source of data (sensors, GPS, database, . . .), and the type of communication to be used to gather coordinates (SMS, MMS, monitoring network, . . .). For this particular action there are no preconditions. Each source has different associated cost and time values, represented by the *source-cost* and *source-time* functions. We only show cost and time for simplicity.

```
(:action get_coordinates
   :parameters (?u - user ?z -zone ?m - source ?c - communication)
   :effect (and (coordinates_taken ?z) (sensed ?z)
   (increase (time) (source-time ?m))
   (increase (cost) (source-cost ?m)) . . .))
```

The *generate_map_*from_vector map action has similar parameters: the user that performs the composition, the zone to be analyzed and the type of map to retrieve from the Internet (maps from NASA, free maps, maps by subscription). Some of the map services allow a limited number of accesses. Therefore, each access decreases the number of available accesses. Consequently, when the number of services reaches zero it is necessary to update the subscription using the *update_subscription* action.

```
(:action generate_vector_map
  :parameters (?usr - user ?z - zone ?mt - maptype)
  :precondition (and (sensed ?z) (> (access ?mt) 0))
  :effect (and (generated_map ?z)
  (increase (time) (/ (time-map ?mt) 2))
  (increase (cost) (/ (cost-map ?mt) 2))
  (decrease (access ?mt) (access_quote))))
```

5. Request analysis module

The Natural language analysis module receives as input the user request done from his/her mobile device in NL, and generates a planning instance in PDDL that represents the state and the goals of the user. The Request Analysis Module is composed of two sub-modules: the sub-module NL Analysis, which parses and processes the user request in natural language; and the sub-module Context Analysis, which takes into account user preferences, device capabilities, and situation (either from some special words or from specific databases) and completes the request with them. Every time a user makes a request a PDDL problem is generated, although the domain in PDDL is always the same.

Formally, let Q be the informal user request expressed in natural language that represents the desired result of the user before its analysis. A user request Q is then defined as $Q = (F, C)$, where F represents the functional part of the request and C represents the context of the query (device, network, situations). In turn, F is specified as an n-tuple $F = \{s_1, s_2, \ldots, s_n\}$. Each s_i denotes one goal or objective and is represented by a binary tuple or pair $\langle G,P \rangle$ where G denotes a verb associated with a goal or objective in PDDL and P denotes a set parameters provided by the user.

These parameters can be classified in Variables V, Places L and Persons P. Examples of Variables are: "temperature", "evacuation route", "stream flow". Among Places we can find: "Cauca River", "my location", "here", "Popayan Bridge". And finally, in the set of Persons λ can be found words such as: "me", "Juan Díaz". A parameter can only be either a variable, a place or a person; formally $V \cap L = \emptyset$, $V \cap P = \emptyset$, and $P \cap L = \emptyset$.

To illustrate this, an example following the previous model is presented here. Let us assume that Q is a request made by an environmental manager from a cell phone. In particular, the request in natural language is *"I need the hydrological balance of zone one"*. In this case, we can infer the following information:

C: The user device is a Nokia Lumia 810 using a 3G Network with HSDPA 900, there are not emergency words in the request.
F: "I need the hydrological balance of zone one".
From F we can extract:
s1: "I need the hydrological balance of zone one"
Analyzing $s1 = \langle G, \{p1, p2, p3\} \rangle$ we obtain:
G = "Calculate"
$p1 \in L$ = "*Zone one*" (A system variable, geo-coded location or a set of coordinates)
$p2 \in P$ = "*me*"
$p3 \in V$ = "*hydrological balance*"

5.1. Context analysis

The context analysis module classifies the information according to three dimensions: *Device*, *User Profile* and *Situation*. Each one of them obtains the information from different sources and defines the selection of domains.

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)[3] and the Composite Capability/Preference Profiles (CC/PP).[4] 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 location of the user can be established using the wireless antenna in the device. Most of the mobile phones can reach the signal of diverse mobile cells, wireless networks or even the devices can incorporate a GPS. So, we exploit the fact that the strength of the signals that a device will receive from different points will vary with location. We build a database of signal strength information for various locations, and use this information to determine the location of the user. This approach is based in its nature in the work of Saha, Chaudhuri, Sanghi, and Bhagwat (2003).

The relationship between the dimensions is established using a preferences function, which assigns weights to each one of them. For instance, consider 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.

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. Table 2 shows how these user criteria are mapped to service properties.

5.2. From natural language to PDDL

The main function of Request Analysis module is to translate the request into a planning instance so the Automated Planning module can use it to perform the composition of services. Like most Automated Planning systems, PELEA takes two inputs expressed in PDDL: a domain description and a problem description. As AUTO is designed to work with a specific domain, the Early Warning Environmental domain, the definition of the domain is static and common to all the planning instances. This means that only the problem file must be generated every time a user request is needed. Due to this the Request Analysis Module focuses on translating the user request and the user context into a planning problem file.

Since the aim of AUTO is to provide the user with a framework for specifying the goal and not the workflow, identifying the relationship between objectives and goals and services and actions is essential. Regarding this, in the Automated Planning module services are represented univocally by actions. Actions contain parameters, preconditions and effects in a similar fashion as the semantic description of Web Services, which makes defining actions in the PDDL domain a straightforward matter in most cases (Table 3).

The linguistic processing of the request in natural language is composed by a set of sequential steps that perform the following tasks: text segmentation, required to separate the words in the phrase; spell checking of the individual words; removing stop

[3] http://wurfl.sourceforge.net/.
[4] http://www.w3.org/Mobile/CCPP/.

Table 1
Context user information.

	User criteria	Values
User context	Network	GPRS/WLAN/GSM
	Device	Cell phone, Laptop
	Location	Outdoor, indoor
User preferences	Data subscription	Yes/No
	Only Free services	Yes/No
	Voice subscription	Yes/No
	Delivery quality	Low, medium, high

Table 2
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

Table 3
Mapping User Request to PDDL.

User criteria	Values
Functional Words + Parameters	Planning Goals
Situational Words	Planning Metrics
Location and device information	Initial state
User preferences	Planning Metrics

words that are considered to be irrelevant (e.g. *want*, *the*, *to*); stemming (e.g. *checks* becomes *check)*; and filtering and tagging of individual units according to their grammatical category. Subsequently Named Entity Recognition performs a classification between "*Situational*", "*Parameters*" and "*Functional*" words based on to their meaning. In order to do so, this module identifies the correct sense of words according to their context, i.e. identifies the correct meaning from a word within multiple meanings that can occur in a sentence.

As described before, the parameter words can be classified as Variables, Places or Persons. Before the translation into PDDL, some parameters are checked in the contact list or other local repositories. For example, "*Mark*" is searched and his ID, in this case his phone number, replaces the occurrences of "*Mark*" in the request. Similarly "Zone 1" is replaced by the coordinates of the zone. Next, information gathered from the User Profile repository using the user's ID is added to the request. Finally, the set of Tokens is translated into PDDL predicates using a regular matching function. The functional tokens and parameters are included as predicates that have to be achieved, while the user preferences are encoded as a linear combination of the relevant criteria.

In order to build a list of potential goals and metrics, we selected the most common important Telecom services used in the Environmental Early Warning domain (Haddow, Bullock, & Coppola, 2007). Then, a repository of the goals associated with these services was created. Next, the repository was enriched with information including *Variables*, *Places* and *Persons*. The metrics are in our case time and cost, as these are the priorities of the users in most cases.

The Advance Telecom services that were selected in the Early Management Warning domain are: "*inform*", "*locate*" and "*calculate*". "*Inform*" is associated with the establishment of a communication through SMS, Voice Call or MMS. "*Calculate*" makes reference to the use of Web services for map generation or specialized environmental algorithms. Meanwhile, "*Locate*" implies the computation of the user position using telecom features, like guessing his/her location by triangulating the position using the coordinates of nearby base stations.

To associate words in a user request to the goals and metrics in PDDL, a chunking-based approach is used. *Chunking* or *shallow parsing* aims at labeling segments of a sentence with syntactic constituents such as *noun* or *verb phrases* (*NP* or *VP* respectively). Each word is assigned only one unique tag, often encoded as a *begin-chunk* (e.g. B-NP) or *inside-chunk* tag (e.g. I-NP). Regarding the precision of such techniques, recent evaluation of chunking-based tools has shown very promising results (Collobert et al., 2011).

An example of user request can be: "*contact Mark urgently*". The resulting PDDL problem file containing the planning objects and metrics would be the following:

```
(:objects
  573006773688 – user;; Mark's phone
  sms mms voice – communication
  google nasa esri – mapservices ...)

(:init
  (=(time) 0) (=(cost) 0) (=(topup_cost) 1)
  (=(sorce-time sms) 10) (=(source-time gps) 5)
  (=(source-cost sms) 5) (=(source-cost sms) 8) ...)
```

The goals are included as predicates that have to be achieved. The user preferences are encoded as a linear combination of the relevant criteria, namely *time* and *cost*. Both *time* and *cost* are multiplied by weights obtained from the user preferences and the context of the request.

```
(:goal (and (informed Mark)))
  (:metric minimize (+ (*(t-weight) (time)) (*(c-weight)
  (cost)))))
```

An alternative to the definition of the metric as a linear combination of two or more functions is its definition as soft goals. In fact, this is the standard way of encoding preferences in PDDL3. However, this offers no additional expressive power over a regular definition of metrics (Keyder & Geffner, 2009) and to date only a few domain independent planners support soft goals, which is why the current definition is better suited to the domain.

6. Prototype and experimentation

In this section a more detailed description of the implementation is given. Also in order to assess the viability of AUTO some experimentation regarding both performance and accuracy will be done. As the AUTO framework is composed of different modules, the experimentation was conducted for each module by separate. Here, we present the results for User Request Analysis and the Automated Planning. The software associated with AUTO Framework including the remaining modules source and binaries are available in the project site.[5]

6.1. User request analysis

The NL Analysis module processes the natural language request through a sequence of steps. Using an annotation based information extraction system called GATE,[6] which offers a Java Based architecture that allows plugging in several kinds of NLP software,

[5] http://www.unicauca.edu.co/~jaordonez/auto.
[6] http://gate.ac.uk/.

such as taggers, sentence splitters and named entity recognizers. In the actual implementation the phrases and plug-ins are configured to deal with Spanish Language. However, for sake of illustrating the functionality this section describes the phrases in English. As an example to show the overall functioning of the module the following common phrases are analyzed.

"I need urgently the evacuation route"
"Send me the Stream flow of Cauca River"

First, the input is split into tokens, generating simple lexical units from complex sentences. Next, words are tagged according to their grammatical nature (Verb, Noun) and their spelling is corrected. After this step the Chunker Pluggin for Spanish language is applied, which labels the segments of the sentences identifying noun phrases such as "my house". Next, the Gazetter and the Transducer module are applied. The Gazetter holds a dictionary with domain dependent words (Environmental management) and the type of the words. The Transducer holds a set of rules for identifying domain dependent words, such as Variables, Places and persons. The Transducer is made up of rules such as the one shown below:

```
Rule: RiversRule
(
{Lookup.minorType == place} // The first word is a place
{Token.string == "river"} // The second word is "river"
):match
-->
:match.river = {rule = "RiversRule"} //label as a river
```

After using the Gazetteer and Transducer, the sentence is split into a set of classified words as shown below in Table 4:

Next, the mapping function is applied to generate PDDL predicates as shown in Table 5.

We will present now two examples in which a given phrase introduced by an user is transformed into goals and metrics in PDDL. The user makes the requests from a Nokia 1100 cell phone first and from a Samsung Galaxy W smartphone second.

Request 1:

"necesito conocer rápidamente las rutas de evacuación"
(I need to know urgently the evacuations routes)

The information extracted from the context is translated to predicates as shown in Table 6.

Table 4
Classification of Input Words.

Word	Classification
Cauca River	Place
Evacuation route	Parameter, Variable
Stream flow	Parameter, Variable

Table 5
Mapping between input tokens and PDDL.

Phrase	Generated PDDL
I need	(Informed me)
Send me	(Informed me)
Evacuation route	(Calculated evac_route my_loc)
Stream flow of Cauca River	(Calculated stream_flow cauca_river)
Urgently	(:metric maximize (time))
I need	(Informed me)

Table 6
Translation of request 1 into PDDL.

Context	Predicates
Phone Nokia 1100: Support for 2G Networks, monochrome display, only supports SMS Messaging	(output sms)
Request	Predicates
Necesito conocer	(Informed me)
Rutas de evacuacion	(Calculated evacuation_route my_location)
Rápidamente	(:metric minimize (time))

For brevity we will only include the most significant parts of the problem instance, that is, goals and metrics. The generated goals and metrics are as follows:

(:goals (and (Informed me) (Calculated evacuation_route my_location) (output sms)))
(:metric minimize (time))

The expected output generated by an expert is the same:

(:goals (and (Informed me) (Calculated evacuation_route my_location) (output sms)))
(:metric minimize (time))

This means that in this case the NL Analysis module was able to automatically translate in a correct way the request in NL into the corresponding problem in PDDL.

Request 2:

"¿Cuál es el refugio más cercano?"
(What is the closest shelter?)

Again, the same process is repeated, as displayed in Table 7. The output of the system is:

(:goals (and (Informed me) (Calculated evacuation_route my_location) (output mms)))
(:metric minimize (cost))

The expected output generated by an expert is:

(:goals (and (Informed me) (Calculated refuges_list my_location) (output mms)))
(:metric minimize (cost))

In this example the system fails to recognize that the user requests only a list of shelters with a single element, the closest one, and instead request the route to the closest shelter. In practice this would not be a big inconvenience, as both requests are similar, although it is not the expected outcome.

Table 7
Translation of request 2 into PDDL.

Context	Predicates
Samsung Galaxy W Support for 3G Networks, supports SMS(threaded view), MMS, Email, Push Mail, IM, RSS	(output mms)
Request	Predicates
-No subject-	(Informed me)
Refugio más cercano	(Calculated evacuation_route my_location)
-no situation information-	(:metric minimize (cost))

As there may be cases like the precedent one, in which the system translates erroneously a part of the request, we conducted a user study to check the accuracy of the translation. The aim of the study was to quantify the difference between human choices and system choices. To achieve this, we gave every subject a hypothetic scenario of flooding in the southern area of Cauca – Colombia, one of the areas with the largest amount of rivers in the country. Subjects write down a list of 10 phrases that could be used in such episode. The list of phrases was processed by the system, and the generated PDDL files were further studied in order to determine if the planning problem could solve the original user request or not.

The sample for our study was composed by individuals from both genders. The sample was divided in two groups: the first group is formed by 10 experts in Early Warnings from the Environmental Studies Group of University of Cauca; the 10 remaining subjects were farmers from the zone. This division was done because the system is designed to interact with both technical users and non-expert users that can be potentially affected in a disaster episode. The use cases were executed on an Intel Pentiun Dual Core 2.2 GHz processor, equipped with 4 GB of RAM. The environment was executed using an instance of GATE 7.0, MySQL 5.1 and JDK 1.6–07.

In total 200 phrases were analyzed, 10 per person. Next we removed repeated phrases and phrases that could not be processed by the system due to the lack of relevance in Early Warning events, e.g. "*when will it rain again?*". After the latter process we obtain 144 phrases.

Since AUTO is designed to be used in real life scenarios, performance is a also an important priority. The first quantitative evaluation focused on the average execution time for analyzing the phrases and generating the PDDL problem file. For the analyzed requests the average time was 0,289 s per processed phrase. Although this cannot be considered a real time response, the average time is significantly low compared to the planning and execution process in most cases.

In terms of accuracy, in order to analyze the quality of the translation a second evaluation was made. This evaluation measures *precision*, *recall* and *F-measure*. Each phrase was translated by hand to the correct set of predicates, describing goals and metrics that could answer to the user request. Subsequently the same phrases were processed using the AUTO Request analysis module. Finally the following measures were computed:

$$\text{Precision(P)} = \frac{\text{Correct}}{\text{Correct} + \text{Spurious}} \tag{1}$$

$$\text{Recall(R)} = \frac{\text{Correct}}{\text{Correct} + \text{Missing}} \tag{2}$$

$$\text{F} - \text{Measure} = \frac{\text{P} * \text{R}}{\text{P} + R} \tag{3}$$

Correct = PDDL predicates generated correctly.
Spurious = PDDL predicates not generated correctly.
Missing = PDDL predicates not generated from the user request that should have appeared.

In this experiment the average precision was 94,409% and the recall 86,736%. The *F-Measure* was 0,448. These results show that the precision is higher that the recall. This due to the fact that recall is associated with missing predicates. It is common that in emergency scenarios, people use special words and not technical words, so which increases the difficulty of the mapping process and causes some key words to be discarded as irrelevant. Nevertheless the translation processes is most of the time accurate enough for the composition process to obtain a service composition adequate to the user request.

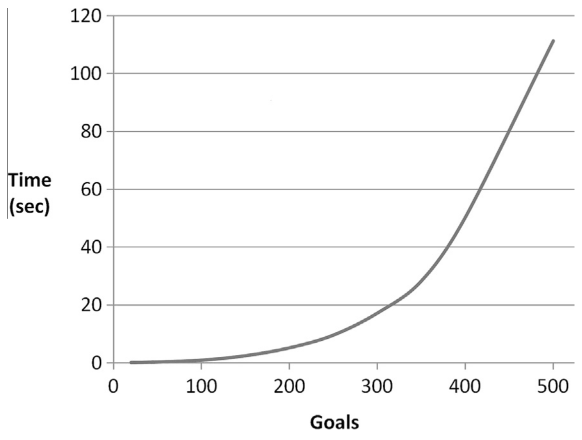

Fig. 4. Time spent by the planner in the service composition.

6.2. Automated planning

To check the viability of the use of automated planning in AUTO we checked whether the system was able to scale up to requests that involve a high number of goals and services. An automatically generated instance was modified to increase the number of conflicting goals[7] (in this case, farms that must be warned using communication means whose costs increase as more farms are warned). The number of goals was increased from 20 to 500 in intervals of 50. Services, communication means, map types, ... were left unmodified. The metric used for these problems was to minimize time plus cost. The planner used was CBP (Fuentetaja, Borrajo, & López, 2010), a planner based on Metric-FF (Hoffmann, 2003) with new heuristics designed to deal with numeric metrics.

Fig. 4 shows how the time increases with the number of goals. Scaling in Automated Planning is a complicated task because of the inherent difficulty of the problems. In fact, domain independent planners need an exponential amount of time on the size of the task to find a solution. In the experimentation, CBP behaves as expected, as the time increases exponentially. However, CBP is still capable of finding a plan with 500 goals in less than 2 min while minimizing the metric. This is a performance competitive with *ad-hoc* algorithms for services composition (Pistore, Traverso, Bertoli, & Marconi, 2005), which confirms the usefulness of domain independent planners for this kind of approaches.

Another of the advantages of Automatic Planning is the possibility of dealing with different metrics just by changing the definition of the metric in the problem. User preferences may be complex and a given user may desire to be able to choose among different compositions of services to solve a given task. This is akin to a multi-objective optimization problem and requires a fair degree of flexibility in the computation of the composition. In our case we tried different weights in the linear combination of functions that compose the metric of a given problem. In particular, if the metric is defined as $M = f(cost,\ time) = \alpha \times cost + \beta \times time$, the weights α and β can be modified to obtain solutions that minimize *time* and *cost* with a different priority. Fig. 5 shows how for the same problem and tweaking the weights α and β in a simple way a set of solutions that form a Pareto front can be computed.

[7] Conflicting goals in Automated Planning are goals such that achieving one makes harder (or costlier) to achieve the rest. Conflicting goals are necessary to increase the difficulty of the problems, as otherwise the planning problem would be linearly separable into several smaller ones, one per goal.

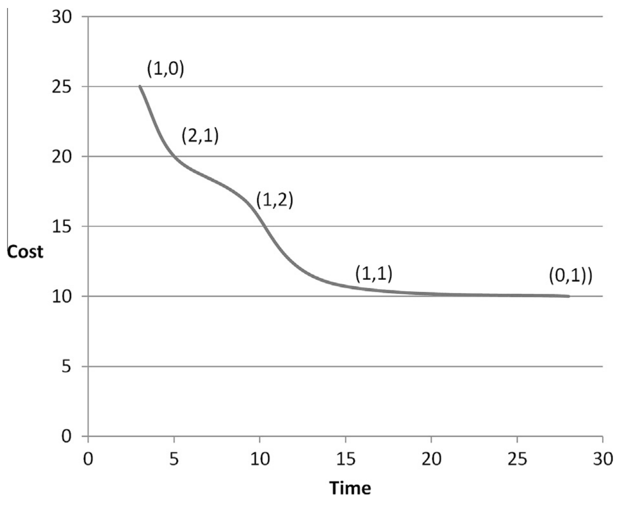

Fig. 5. Cost and time of the different solutions obtained by modifying α and β (between parenthesis, respectively).

7. Related work

Zhu, Zhang, Chen, and Cheng (2010) presented Hybrid Service Creation and Execution Environment (HSCEE), a Template-based service creation platform with low latency service execution. HSCEE is based in on BPEL templates but most of the design tasks must be done manually. Natural language processing is also not considered.

The OPUCE Project[8] (Yu, Sheng, Han, Wu, & Liu, 2012) (Yelmo, del Álamo, Trapero, & Martín, 2011) presented a User-centric service creation environment (SCE) for the composition of Web and telecom services. This SCE includes a semantic browser, a user-centric environment and a translator to Web Services Business Process Execution Language (WS-BPEL Version 2.0, 2007). These works are focused on the use of semantic technologies for service description and discovery; however they do not consider natural languages interfaces and the composition is not done automatically.

Kim, Kang, Meadows, Ioup, and Sample (2009) presented a framework for automatic composition of services, focusing only on web domains without concern for the execution phase. da Silva, Pires, and van Sinderen (2011) proposed frameworks for automatic composition exploiting natural language processing and semantic annotations for service matchmaking based on SPATEL language. They do not address the validation of the non-functional properties and are focused only on the request analysis and plan generation. Our approach deals with all the phases including execution and reconfiguration based on JSLEE environments.

Kaldeli, Lazovik, and Aiello (2011) present an approach for Web service composition based on Automated Planning with preferences. They present an algorithm for interleaving planning, monitoring and execution of services. This approach uses continual planning based in the data extracted of continual feedback of the execution.

Finally, Hatzi et al. (2012) proposed the use of Automated Planning to address automated web service composition. The authors present a semantic-based technique for discovery and approximate composition of services; furthermore, this approach shows experiments about the obtained composition of services in terms of accuracy. However, this work does not consider convergent scenarios with Web and Telco services and does not provide a natural language interface for accessing the system.

[8] http://www.opuce.eu/.

Table 8
Analysis of Related work.

Works	Include all phases	User centered	Consider convergence
(Zu et al., 2010) (Yu et al., 2012) (Yelmo et al., 2011)	Yes, but not all of them are automatic or natural language based.	Yes	Yes
(Kim et al., 2009)	No, only the request analysis and the service creation	No	No, focused on the web domain
(da silva et al., 2011)	No, just the request analysis and the service creation	No	Yes
(Kaldeli et al., 2011)	No, just the OWL-S based planning	No	No, focused on the Web domain
(Hatzi et al., 2012)	No, just the planning based composition and discovery	No	No

Table 8 outlines a comparison of related work based on the following criteria: first, if the work deals with all the phases of the service composition process including discovery, composition, execution and reconfiguration. Second, if the approach for service composition is user-centered. Third, if the approach takes into account convergence of services.

8. Conclusion

In this paper we extend AUTO by integrating Automated Planning into an existing architecture as the deliberative process. This is a more general approach than the previously used technique that allows a greater flexibility both in terms of design and the implementation of the underlying algorithms. The main contribution is the automatic generation of planning instances in PDDL from natural language requests. Furthermore, we allowed the use of preferences modeling them as the metrics of the planning instance. In the near future, we want to continue our research developing mechanisms for the automation of the generation of the planning domain. Further experimentation with different planners will be done too so an increase in performance can be achieved. Additionally, we are extending the preferences criteria in order to get a better personalized experience for the user.

Finally, a complete integration with PELEA will be done, so features that it already offers, like monitoring and replanning, can be used by AUTO. Future works include the development of mechanisms for automation of deployment of services in Telecommunication environments. These environments will provide more reliability in order to support a wide amount of telecommunication services and provide a higher reliability.

References

Carman, M., Serafini, L., & Traverso, P., (2003, June). Web service composition as planning. In *ICAPS 2003 workshop on planning for web services* (pp. 1636–1642).

Catarci, T., de Leoni, M., Marrella, A., Mecella, M., Russo, A., Steinmann, R., et al. (2011). Workpad: process management and geo-collaboration help disaster response. *International Journal of Information Systems for Crisis Response and Management (IJISCRAM), 3*(1), 32–49.

Chrighton, C., Long, D. T., & Page, D. C., 2007. JAIN SLEE vs SIP Servlet Which is the best choice for an IMS application server? In *Proceedings of 2007 Australasian telecommunication networks and applications conference*. doi: 10.1109/ATNAC.2007.4665289.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research, 12*, 2493–2537.

Cremene, M., Tigli, J. Y., Lavirotte, S., Pop, F. C., Riveill, M., & Rey, G. (2009). Service composition based on natural language requests. In *IEEE International Conference on Services Computing, SCC'09* (pp. 486–489). IEEE.

da Silva, E. G., Pires, L. F., & van Sinderen, M. (2011). Towards runtime discovery, selection and composition of semantic services. *Computer Communications, 34*(2), 159–168.

Dustdar, S., & Schreiner, W. (2005). A survey on web services composition. *International Journal of Web and Grid Services, 1*(1), 1–30.

Fuentetaja, R., Borrajo, D., & López, C. L. (2010). A look-ahead B&B search for cost-based planning. In *Current Topics in Artificial Intelligence* (pp. 201–211). Berlin, Heidelberg: Springer.

Gerevini, A. E., Haslum, P., Long, D., Saetti, A., & Dimopoulos, Y. (2009). Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence, 173*(5), 619–668.

Guzmán, C., Alcázar, V., Prior, D., Onaindia, E., Borrajo, D., Fdez-Olivares, J., & Quintero, E., (2012). Pelea: a domain-independent architecture for planning, execution and learning. In *ICAPS Workshop on Scheduling and Planning Applications woRKshop (SPARK)* (pp. 38–45).

Haddow, G., Bullock, J., & Coppola, D. P. (2007). *Introduction to emergency management*. Butterworth-Heinemann.

Hatzi, O., Vrakas, D., Nikolaidou, M., Bassiliades, N., Anagnostopoulos, D., & Vlahavas, I. (2012). An integrated approach to automated semantic web service composition through planning. *IEEE Transactions on Services Computing, 5*(3), 319–332.

Hoffmann, J. (2003). The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research (JAIR), 20*, 291–341.

Hoffmann, J., Bertoli, P., & Pistore, M. (2007). Web service composition as planning, revisited: In between background theories and initial state uncertainty. In *Proceedings of the national conference on artificial intelligence* (pp. 1013–1018). Vancouver: AAAI Press.

Huang, C., Lee, G. M., & Crespi, N. (2012). A semantic enhanced service exposure model for a converged service environment. *Communications Magazine, IEEE, 50*(3), 32–40.

Kaldeli, E., Lazovik, A., & Aiello, M. (2011). *Continual planning with sensing for web service composition*. AAAI.

Keyder, E., & Geffner, H. (2009). Soft goals can be compiled away. *Journal of Artificial Intelligence Research, 36*(1), 547–556.

Kim, A., Kang, M., Meadows, C., Ioup, E., & Sample, J., (2009, April). A framework for automatic web service composition. US Naval research Lab., Washington D.C., Tech. Rep. NRL/MR/5540–09-9191.

Laga, N., Bertin, E., Glitho, R., & Crespi, N. (2012). Widgets and composition mechanism for service creation by ordinary users. *Communications Magazine, IEEE, 50*(3), 52–60.

Lim, J., & Lee, K. H. (2010). Constructing composite web services from natural language requests. *Web Semantics: Science, Services and Agents on the World Wide Web, 8*(1), 1–13.

Object Management Group, 2012. *Profile for advanced and integrated telecommunication services*. Retrieved 14.1.2014, from <http://www.omg.org/spec/TelcoML/1.0/Beta1/PDF/>.

Oh, S. C., Lee, D., & Kumara, S. R. (2006). A comparative illustration of AI planning-based web services composition. *ACM SIGecom Exchanges, 5*(5), 1–10.

Pistore, M., Traverso, P., Bertoli, P., & Marconi, A. (2005). Automated synthesis of executable web service compositions from BPEL4WS processes. In *Special interest tracks and posters of the 14th international conference on World Wide Web* (pp. 1186–1187). ACM.

Saha, S., Chaudhuri, K., Sanghi, D., & Bhagwat, P. (2003). Location determination of a mobile device using IEEE 802.11b access point signals. In *Wireless Communications and Networking, WCNC* (3, pp. 1987–1992). IEEE.

Sohrabi, S., Prokoshyna, N., & McIlraith, S. A. (2009). Web service composition via the customization of Golog programs with user preferences. In *Conceptual modeling: foundations and applications* (pp. 319–334). Berlin Heidelberg: Springer.

Ye, Z., Zhou, X., & Bouguettaya, A. (2011). Genetic algorithm based QoS-aware service compositions in cloud computing. In *Database systems for advanced applications* (pp. 321–334). Berlin Heidelberg: Springer.

Yelmo, J. C., del Álamo, J. M., Trapero, R., & Martín, Y. S. (2011). A user-centric approach to service creation and delivery over next generation networks. *Computer Communications, 34*(2), 209–222.

Yu, J., Sheng, Q. Z., Han, J., Wu, Y., & Liu, C. (2012). A semantically enhanced service repository for user-centric service discovery and management. *Data & Knowledge Engineering, 72*, 202–218.

Zhao, Z., Bhattarai, S., Liu, J., & Crespi, N. (2011). Mashup services to daily activities: End-user perspective in designing a consumer mashups. In *Proceedings of the 13th international conference on information integration and web-based applications and services* (pp. 222–229). ACM.

Zhu, D., Zhang, Y., Chen, J., & Cheng, B. (2010). Enhancing ESB based execution platform to support flexible communication web services over heterogeneous networks. In *International conference on communications (ICC)* (pp. 1–6). IEEE.