IDUM a Logic-Based System for e-Tourism

Giuliano Candreva[1], Gianfranco De Franco[2], Dino De Santo[1], Carmine Donato[3], Antonella Dimasi[4], Giovanni Grasso[5], Salvatore Maria Ielpa[5], Salvatore Iiritano[4], Nicola Leone[5], and Francesco Ricca[5]

[1]ASPIdea Software farm, Via Ungaretti 27, 87036 Rende (CS), Italy
giuliano.candreva@infotouch.it
[2]Top Class srl - Travel Agency, Via Busento 21, 87036 Rende (CS), Italy
gianfranco.defranco@gmail.com, dino@topclassturismo.com
[3]Spin scrl - Consorzio di Ricerca in Tecnologie dell'Informazione e della Comunicazione, Via degli Stadi 22/F, 87100 Cosenza, Italy
donato@consorziospin.it
[4]Exeura Srl, Via Pedro Alvares Cabrai - C.da Lecco 87036 Rende (CS), Italy
{antonella.dimasi,salvatore.iiritano}@exeura.com
[5]Dipartimento di Matematica, Università della Calabria, 87030 Rende, Italy
{grasso,s.ielpa,leone,ricca}@mat.unical.it

Abstract. In this paper we present the IDUM system, a successful application of logic programming to e-tourism. IDUM exploits two technologies that are based on the state-of-the-art ASP system DLV: (i) a system for ontology representation and reasoning, called OntoDLV; and, (ii) HiLεX, a semantic information extraction tool. The core of IDUM is an ontology which models the domain of the touristic offers. The ontology is automatically populated by extracting the information contained in the touristic leaflets produced by tour operators. A set of specifically devised logic programs is used to reason on the information contained in the ontology for selecting the holiday packages that best fit the customer needs. An intuitive web-based user interface eases the task of interacting with the system for both the customers and the operators of a travel agency.

1 Introduction

In the last few years, the tourism industry strongly modified the marketing strategies with the diffusion of e-tourism portals in the Internet. Big tour operators are exploiting new technologies, such as web portals and e-mails, for both simplifying their advertising strategies and reducing the selling costs. The efficacy of e-tourism solutions is also witnessed by the continuously growing community of e-buyers that prefer to surf the Internet for buying holiday packages. On the other hand, traditional travel agencies are undergoing a progressive lost of marketing competitiveness. This is partially due to the presence of web portals, which basically exploit a new market. Indeed, Internet surfers often like to be engaged in self-composing their holiday by manually searching for flights, accommodation etc. Instead, the traditional selling process, which has its strength on both the direct contact with the customer and the knowledge about the customer habits, is experiencing a reduced efficiency. This can be explained by the increased complexity of matching demand and offer. Indeed, travel agencies receive

thousand of e-mails per day from tour operators containing new pre-packaged offers. Consequently, the employees of the agency cannot know all the available holiday packages (since they cannot analyze all of them). Moreover, customers are more exigent than in the past (e.g. the classic statement "I like the sea" might be enriched by "I like snorkeling", or "please find an hotel in Cortina" might be followed by "featuring beauty and fitness center") and they often do not declare immediately all their preferences and/or constraints (like, e.g., budget limits, preferred transportation mean or accommodation etc.). This knowledge of customers preferences plays a central role in the traditional selling process, but matching this information with the large unstructured e-mail database is both difficult to be carried out in a precise way and time consuming. Consequently, the seller is often unable to find in a short time the best possible solution.

The goal of the IDUM project is to devise a system that addresses above-mentioned causes of inefficiency by offering:

> (i) an automatic extraction and classification of the incoming touristic offers (so that they are immediately available for the seller), and
> (ii) an "intelligent" search that combines knowledge about users preferences with geographical information, and matches user needs with the available offers.

We reach the goal by exploiting computational logic, and, in particular, Answer Set Programming (ASP) [1]. ASP is a powerful logic programming language, which is very expressive in a precise mathematical sense; in its general form, allowing for disjunction in rule heads and nonmonotonic negation in rule bodies, ASP can represent in a fully declarative way *every* problem in the complexity class Σ_2^{P} and Π_2^{P} (under brave and cautious reasoning, respectively) [2]. In particular, the core functionalities of IDUM are based on two technologies[1] relying on the state-of-the-art ASP system DLV [3]:

- OntoDLV [4, 5] a powerful ASP-based ontology representation and reasoning system; and,
- HiLeX [6–8], an advanced tool for semantic information-extraction from unstructured or semi-structured documents.

More in detail, in the IDUM system, behind the web-based user interface (that can be used by both employees of the agency and customers), there is an "intelligent" core that exploits an OntoDLP ontology for both modeling the domain of discourse (i.e., geographic information, user preferences, and touristic offers, etc.) and storing the available data. The ontology is automatically populated by extracting the information contained in the touristic leaflets produced by tour operators. It is worth noting that offers are mostly received by the travel agency in a dedicated e-mail account. Moreover, the received e-mails are human-readable, and the details are often contained in email-attachments of different format (plain text, pdf, gif, or jpeg files) and structure that might contain a mix of text and images. The HiLeX system allows for automatically processing the received contents, and to populate the ontology with the data extracted

[1] Both systems are developed by Exeura srl, a technology company working on analytics, data mining, and knowledge management, that is working on their industrialization finalized to commercial distribution.

from touristic leaflets. Once the information is loaded on the ontology, the user can perform an "intelligent" search for selecting the holiday packages that best fit the customer needs. IDUM tries to mimic the behavior of the typical employee of a travel agency by exploiting a set of specifically devised logic programs that "reason" on the information contained in the ontology.

In the remainder of the paper, we first introduce the employed ASP-based technologies; then, in Section 3, we describe how the crucial tasks have been implemented. We show the architecture of the IDUM system in Section 4, and we draw the conclusion in Section 6.

2 Underlying Logic-based technologies

The core functionalities of the e-tourism systems IDUM were based on two technologies relying on the DLV system [3]: OntoDLV [4, 5] a powerful ASP-based ontology representation and reasoning system; and, HiLEX [6–8], an advanced tool for semantic information-extraction from unstructured or semi-structured documents.

2.1 The OntoDLV System

Traditional ASP in not well-suited for ontology specifications, since it does not directly support features like classes, taxonomies, individuals, etc. Moreover, ASP systems are a long way from comfortably enabling the development of industry-level applications, mainly because they lack important tools for supporting programmers. All the above-mentioned issues were addressed in OntoDLV [4, 5], a system for ontologies specification and reasoning. Indeed, by using OntoDLV, domain experts can create, modify, store, navigate, and query ontologies; and, at the same time, application developers can easily develop their own knowledge-based applications on top of OntoDLV, by exploiting a complete Application Programming Interface [9]. OntoDLV implements a powerful logic-based ontology representation language, called OntoDLP, which is an extension of (disjunctive) ASP with all the main ontology constructs including classes, inheritance, relations, and axioms. In OntoDLP, a *class* can be thought of as a collection of individuals that share some features. An individual, or *object*, is any identifiable entity in the universe of discourse. Objects, also called class instances, are unambiguously identified by their object-identifier (oid) and belong to a class. A class is defined by a name (which is unique) and an ordered list of attributes, identifying the properties of its instances. Each attribute has a name and a type, which is, in truth, a class. This allows for the specification of *complex objects* (objects made of other objects). Classes can be organized in a specialization hierarchy (or data-type taxonomy) using the built-in *is-a* relation (*multiple inheritance*). Relationships among objects are represented by means of *relations*, which, like classes, are defined by a (unique) name and an ordered list of attributes (with name and type). OntoDLP relations are strongly typed while in ASP relations are represented by means of simple flat predicates. Importantly, OntoDLP supports two kind of classes and relations: (base) classes and (base) relations, that correspond to basic facts (that can be stored in a database); and *collection* classes and *intensional* relations, that correspond to facts that can be inferred by logic programs; in particular, *collection classes* are mainly intended for object reclassification (i.e., for classifying individuals of an ontology repeatedly). For instance, the following

statement declares a class modeling customers, which has six attributes, namely: *firstName*, *lastName*, and *status* of type string; *birthDate* of type Date; a positive integer *childNumber*, and *job* which contains an instance of another class called Job.

```
class Customer (firstName: string, lastName: string,
    birthDate: Date, status: string,
    childNumber: positive integer, job: Job).
```

As in ASP, logic programs are sets of logic rules and constraints. However, OntoDLP extends the definition of logic atom by introducing class and relation predicates, and complex terms (allowing for a direct access to object properties). This way, the OntoDLP rules merge, in a simple and natural way, the declarative style of logic programming with the navigational style of the object-oriented systems. In addition, logic programs are organized in *reasoning modules*, to take advantage of the benefits of modular programming. For example, with the following program we single out the pairs of customers having the same age

```
module (CustomersWithTheSameAge) {
  sameAge(C1,C2,D) :- C1: Customer(birthDate:D),
                      C2: Customer(birthDate:D).
}
```

The core of OntoDLV is a rewriting procedure [5] that translates ontologies, and reasoning modules to an equivalent standard ASP program which, in the general case, runs on state-of-the art ASP system DLV [3].

OntoDLV features an advanced persistency manager that allows one to store ontologies transparently both in text files and internal relational databases; furthermore, powerful type-checking routines are able to analyze ontology specifications and single out consistency problems. Importantly, if the rewritten program is stratified and non disjunctive [1, 10, 11] (and the input ontology resides in relational databases) the evaluation is carried out directly in mass memory by exploiting a specialized version of the same system, called DLV ^{DB} [17]. Note that, since class and relation specifications are rewritten into stratified and non-disjunctive programs, queries on ontologies can always be evaluated by exploiting a DBMS. This makes the evaluation process very efficient, and allows the knowledge engineer to formulate queries in a language more expressive than SQL. Clearly, more complex reasoning tasks (whose complexity is NP/co-NP, and up to Σ_2^P/Π_2^P) are dealt with by exploiting the standard DLV system instead.

2.2 The HiLeX System

HiLeX [6–8] is an advanced system for ontology-based information extraction from semi-structured and unstructured documents, that has been already exploited in many relevant real-world applications. The HiLeX system implements a semantic approach to the information extraction problem based on a new-generation semantic conceptual model exploiting:

- ontologies as knowledge representation formalism;
- a general document representation model able to unify different document formats (html, pdf, doc, ...);

– the definition of a formal attribute grammar able to describe, by means of declarative rules, objects/classes w.r.t. a given ontology.

Most of the existing approaches do not work in a semantical way and they are not independent from the particular type of document they process. Contrariwise, the approach implemented in HiLεX confirms that it is possible recognize, extract and structure relevant information form heterogeneous sources.

HiLεX is based on OntoDLP for describing ontologies, because this language perfectly fits the definition of semantic extraction rules.

Regarding the unified document representation, the idea is that a document (unstructured or semi-structured) can be seen as a suitable arrangement of objects in a two-dimensional space. Each object has an own semantics, is characterized by some attributes and it is located in a two-dimensional area of the document called *portion*. A portion is defined as a rectangular area univocally identified by four cartesian coordinates of two opposite vertices. Each portion "contains" one or more objects and an object can be recognized in different portions.

The language of HiLεX is founded on the concept of *ontology descriptor*. A "descriptor" looks like a production rule in a formal attribute grammar, where syntactic items are replaced by ontology elements, and where extensions for managing two-dimensional objects are added. Each descriptor allows to describe: (i) an ontology object in order to recognize it in a document; or (ii) how to "generate" a new object that, in turn, may be added to the original ontology.

In the following example, we report a part of an ontology and two HiLεX descriptors that identify the instances `'sunny'` (or `'windy'`) of the class `weatherTerm`.

```
// core ontology (provided by the system)
class hiToken (parent:hiBox, value:string).
class hiBox (paragraph:[hiToken]).
relation hasLemma (token:hiToken, lemma:string, tag:hiPosTag).
// domain ontology (provided by the user)
class weatherTerm (descr:string).
'sunny': weatherTerm (descr:"Sunny").
'wind: weatherTerm (descr:"Windy").

// descriptors for existing objects
<'sunny'> -> <T:hiToken(), hasLemma(token:T, lemma:"sunny")>.
<'windy'> -> <T:hiToken(), hasLemma(token:T, lemma:"wind")>.
```

Here, an instance of `weatherTerm` is discovered in a document if the latter contains a portion associated to a basic object of type *hiToken* (collecting basic objects that represent all the substrings) satisfying the condition that its lemma is `"sunny"` (or `"windy"`). Note that an object may also have more than one descriptor, thus allowing one to recognize the same kind of information when it is presented in different ways.

The HiLεX system has been applied to different real-world problems [6–8] such as: (i) Information extraction from balance sheets; (ii) Information extraction from news; (iii) Information extraction from clinical documents.

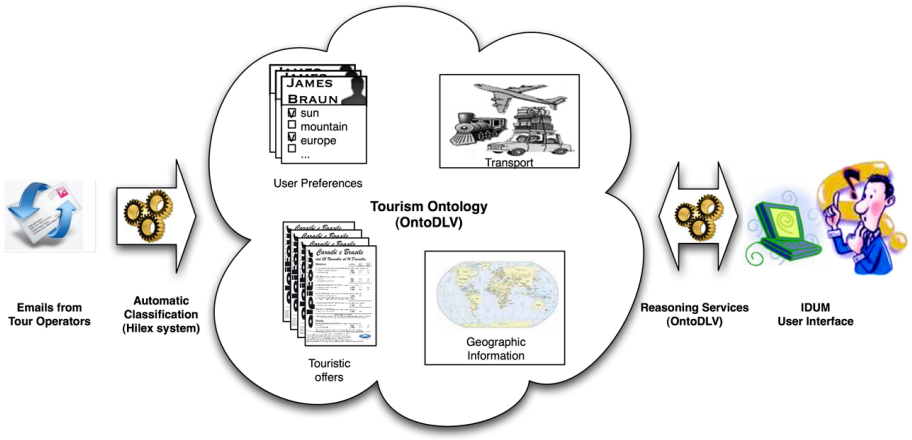

Fig. 1. The IDUM System.

3 The IDUM System

In this section we describe the core of the IDUM system and its innovative features based on ASP. IDUM is an e-tourism system, conceived for classifying and driving the search of touristic offers for both travel agencies operators and their customers.

IDUM, like other existing portals, has been equipped with a proper (web-based) user interface; but, behind the user interface there is an "intelligent" core that exploits knowledge representation and reasoning technologies based on ASP. In IDUM (see Figure 1), the information regarding the touristic offers provided by tour operators is received by the system as a set of e-mails. Each e-mail might contain plain text and/or a set of leaflets, usually distributed as pdf or image files which store the details of the offer (e.g., place, accommodation, price etc.). Leaflets are devised to be human-readable, might contain both text and images, and usually do not have the same structure. E-mails (and their content) are automatically processed by using the HiLeX system, and the extracted data about touristic offers is used to populate an OntoDLP ontology that models the domain of discourse: the "*tourism ontology*". The resulting ontology is then analyzed by exploiting a set of reasoning modules (ASP programs) combining the extracted data with the knowledge regarding places (geographical information) and users (user preferences) already present in the tourism ontology. The system mimics the typical deductions made by a travel agency employee for selecting the most appropriate answers to the user needs.

In the following sections, we briefly describe the tourism ontology and the implementation of the above-mentioned ASP-based features.

3.1 The Tourism Ontology

The "tourism ontology" has been specified by analyzing the nature of the input (we studied the characteristics of several touristic leaflets) with the cooperation of the staff

```
class Customer (firstName: string, lastName: string,
    birthDate: Date, status: string,
    childNumber: positive integer, job: Job).
relation CustomerPrefersTrip ( cust:Customer, kind: TripKind ).
relation CustomerPrefersMeans ( cust:Customer,
    means: TransportationMean ).
...

class Place ( description:string ).
intentional relation Contains ( pl1:place, pl2:place )
{ Contains(P1,P2) :- Contains(P1,P3), Contains(P3,P2).
 Contains('Europe', 'Italy'). Contains('Italy', 'Sicily').
 Contains('Sicily', 'Palermo'). ... }

relation PlaceOffer( place: place, kind: tripKind ).
relation SuggestedPeriod ( place:place, period: positive integer ).
relation BadPeriod ( place:place, period: positive integer ).

class TouristicOffer( start: Place, destination: Place,
    kind: TripKind, means: TransportationMean,
    cost: positive integer, fromDay: Date, toDay: Date,
    maxDuration: positive integer, deadline: Date,
    uri: string, tourOperator: TourOperator ).

class TransportationMean ( description: string ).
class TripKind ( description: string ).
...
```

Fig. 2. Main entities of the touristic ontology.

of a real touristic agency, which has been repeatedly interviewed. In this way, we could model the key entities that describe the process of organizing and selling a complete holiday package. In particular, the "tourism ontology" models all the required information: user profile, geographic information, kind of holiday, transportation means, etc. In Figure 2, we report some of the most relevant classes and relations that constitute the tourism ontology. In detail, the class *Customer* allows one to model the personal information of each customer, while a number of relations is used to model user preferences, like *CustomerPrefersTrip* and *CustomerPrefersMeans*, that associate each customer to its preferred kind of trip, and its preferred transportation means, respectively. The kind of trip is represented by using a class *TripKind*. Examples of *TripKind* instances are: *safari*, *sea_holiday*, etc. In the same way, e.g. *airplane*, *train*, etc. are instances of the class *TransportationMean*. Geographical information is modeled by means of the class *place*, which has been populated with the information regarding more than a thousand of touristic places. Moreover, each place is associated to a kind of trip by means of the relation *PlaceOffer* (e.g. Kenya offers safari, Sicily offers both sea and sightseeing). Importantly, the natural a *part-of* hierarchy of places is easily modeled by using the intensional relation defining an *Contains*. This allowed us to assert manually only some basic facts and to obtain in a simple yet powerful way all the basic inclusions. Indeed,

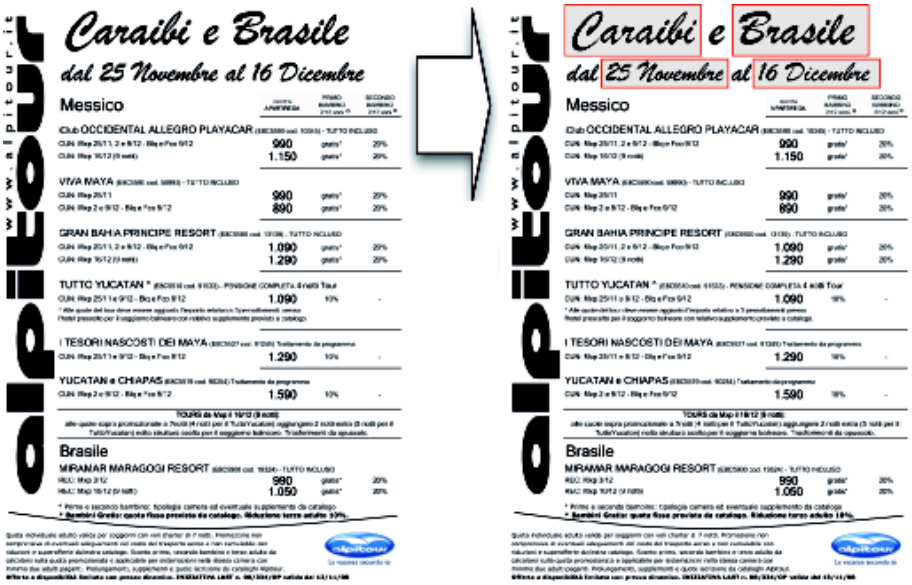

Fig. 3. Extracting offer information.

the full hierarchy is computed by evaluating a rule (that, basically, encodes the transitive closure).

The mere geographic information is, then, enriched by other information that is usually exploited by travel agency employees for selecting a travel destination. For instance, one might suggest to avoid sea holidays in winter, or to go in India during the wet monsoon period; whereas, one should be suggested to visit Sicily in summer. This was encoded by means of the two relations *SuggestedPeriod* and *BadPeriod*.

Finally, the *TouristicOffer* class contains an instance for each available holiday package. The instances of this class are added either automatically, by exploiting the HiLeX system (as described in the next section), or manually by the personnel of the agency.

3.2 Automatic Extraction of Touristic Offers

Touristic offers are mainly available in digital format, and are received via e-mail. It is usual that more than a hundred emails per day crowd the mail folder of a travel agency, and often the personnel cannot even analyze the entire in-box. This causes a loss of efficiency in the selling process, because some interesting offers might be ignored. Note that most of the information is contained in pdf, gif or jpeg files attached to the e-mail messages, and this strongly limits the efficacy of standard search tools like, e.g., the ones provided by e-mail clients.

To deal with this problem, the IDUM system has been equipped with an automatic classification system based on HiLeX. Basically, after some pre-processing steps, in

which e-mails are automatically read from the inbox, and their attachments are properly handled (e.g. image files are analyzed by using OCR software), the input is sent to HiLeX. In turn, HiLeX is able to both extract the information contained in the e-mails and populate the *TouristicOffer* class. This has been obtained by encoding several ontology descriptors (actually, we provided several descriptors for each kind of file received by the agency). For instance, the following descriptor:

```
<TouristicOffer (Destination, Period)> ->
    <X:place(XX)>{Destination:=X;}
    <X:date(XX)>{Period:=X;}
    SEPBY <X:separator()>.
```

allows to extract from the leaflet in Figure 3(a) that the proposed holiday package regards trips to both the Caribbean islands and Brazil. Moreover it also extracts the period in which this trip is offered. The extracted portions are outlined in Figure 3(b). The result of the application of this descriptor are two new instances of the TouristicOffer class.

3.3 Personalized Trip Search

The second crucial task carried out in the IDUM system is the personalized trip search. This feature has been conceived to make simpler the task of selecting the holiday packages that best fit the customer needs. We tried to "simulate" the deductions made by an employee of the travel agency in the selling process by using a set of specifically devised logic programs. In a typical scenario, when a customer enters the travel agency, an employee tries to understand her current desires and preferences at first; then, the seller has to match the obtained information information with a number of pre-packaged offers. Actually, a number of candidate offers are proposed to the customer and, then, the employee has to understand her needs by interpreting the preferences of the customer. Customer preferences depends on her personal information (age, gender, marital status, lifestyle, budget, etc.), but also on her holiday habits (e.g. she prefers going to the mountains, or she has already been in Italy). Actually, this information has to be elicited by the seller by interviewing the customer, but most of it might be already known by the employee if she is serving an old customer. In this process, what has to be clearly understood for selecting an holiday package fitting the customer needs is resumed in the following four words: *where*, *when*, *how*, and *budget*. Indeed, the seller has to understand *where* the customer desires to go; *when* she can/wishes to leave; how much time she will dedicate to his holiday; which is the preferred transportation mean (*how*); and, finally, the available budget. However, the customer does not directly specify all this information, for example, she can ask for a sea holiday in January but she does not specify a precise place, or she can ask for a kind of trip that is unfeasible in a given period (e.g. winter holiday in Italy in August) In this case the seller has to exploit her knowledge of the world for selecting the right destination and a good offer in a huge list of proposals. This is exactly what the IDUM system does when an user starts a new search. Current needs are specified by filling an appropriate search form (see Figure 4), where some of the key information has to be provided (i.e. where and/or when and/or available money and/or how). Note that, the tourism ontology models both the knowledge of the seller (geographic information and preferred places), and the profile of customers (clearly, in the case of new customers the seller has to fill the profile information, whereas the

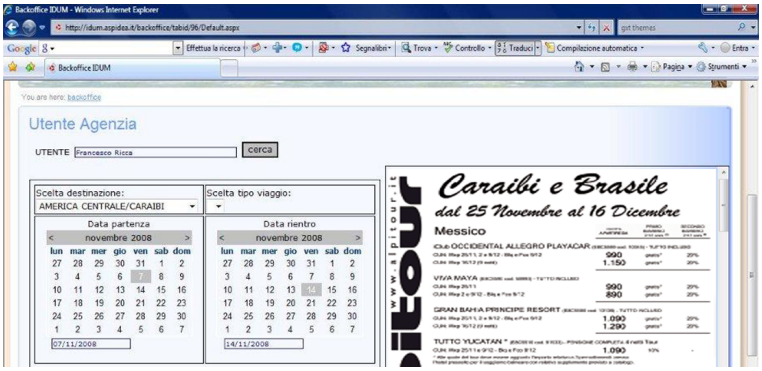

Fig. 4. The IDUM User Interface.

ontology already contains the information regarding old customers); moreover, the extraction process continuously populates the ontology with new touristic offers. Thus, the system, by running a specifically devised reasoning module, combines the specified information with the one available in the ontology, and shows the holiday packages that best fit the customer needs. For example, suppose that a customer specifies the kind of holiday and the period, then the following module selects of holiday packages:

```
module(kindAndPeriod) {
  %detect possible and suggested places
  possiblePlace(P) :- askFor(tripKind:K),
                      PlaceOffer(place:P, kind:K).

  suggestPlace(P) :- possiblePlace(P), askFor(period:D),
                     SuggestedPeriod(place:P1, period:D),
                     not BadPeriod(place:P1, period:D).

  %select possible, alternative and suggestible packages
  possibleOffer(O) :- O:TouristicOffer(destination:P),
                      possiblePlace(P).

  alternativeOffer(O) :- O:TouristicOffer(destination:P),
                         suggestPlace(P).

  suggestOffer(O) :- O:TouristicOffer(destination:P, mean:M),
                     suggestPlace(P), askFor(cust:C),
                     CustomerPrefersMean(cust:C, mean:M).
}
```

The first two rules select: possible places (i.e., the ones that offer the kind of holiday in input); and places to be suggested because they offer the required kind of holiday in the specified period. Finally, the remaining three rules search in the available holiday packages the ones that: offer an holiday that matches the original input (possible offer); are good alternatives in suggested places (alternativeOffer); or match the customer's preferred mean and can be suggested.

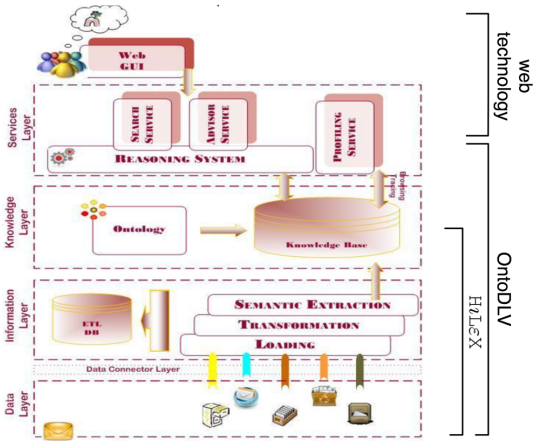

Fig. 5. System Architecture.

4 System Architecture

The architecture of the IDUM system, which is depicted in Figure 5, is made of four layers: *Data Layer, Information Layer, Knowledge Layer*, and *Service Layer*. In the Data Layer the input sources are dealt with. In particular, the system is able to store and handle the most common kind of sources: e-mails, plain text, pdf, gif, jpeg, and HTML files. The Information Layer provides ETL (Extraction, Transformation and Loading) functionalities, in particular: in the loading step the documents to be processed are stored in an auxiliary database (that also manages the information about the state of the extraction activities); whereas, in the Transformation step, semi-structured or non-structured documents are manipulated. First the document format is normalized; then, the "bi-dimensional logical representation" is generated (basically, the HiLeX portions are identified); finally, HiLeX descriptors are applied in the Semantic Extraction step and ontology instances are recognized within processed documents. The outcome of this process is a set of concept instances, that are recognized by matching semantic patterns, and stored in the core knowledge base of the system where the tourism ontology resides (Knowledge Layer). Domain ontology and extracted information are handled by exploiting the Persistency manager of the OntoDLV system (see Section 2.1). The Services Layer features the profiling service and the intelligent search (see Section 3.3) which implements the reasoning on the core ontology by evaluating in the OntoDLV system a set of logic programs. The Graphical User Interface (GUI) can access the system features by interacting with a set of web-services.

5 Related Work

The usage of ontologies for developing e-tourism applications was already studied in the literature [12, 14, 13], and the potential of the application of semantic technology recognized. The architecture of an e-tourism system able to create a touristic package

in a dynamic way is presented in [14]. It is based on an ontology written in OWL-DL. An advantage of the our approach is the possibility of developing ASP programs that reason on the data contained in the ontology for developing complex searches, while there is no accepted solution for combining logic rules and OWL-DL ontologies.

The SPETA system [15], which is based on the ontology of [14], acts as an advisor for tourists. Fundamentally, SPETA follows people who need advising when visiting a new place, and who consequently do not know what is interesting to visit. Both SPETA and IDUM exploit an ontology for building personalized solutions for the users, but the goal of SPETA (offering assistance during travels) is different from that of IDUM.

The E-Tourism Working Group at DERI [16] is developing e-tourism solutions based on the Semantic Web technology. Deri also developed a content management system: OnTourism. The solution is very similar to IDUM, but it is based on the use of Lixto Software which makes information extraction from web pages.

6 Conclusion

In this paper we described a successful example of commercial and practical use of logic programming: the e-tourism system IDUM.

The core of IDUM is an ontology modeling the domain of the touristic offers, which is automatically populated by extracting the information contained in the e-mails sent by tour operators; and an intelligent search tool based on answer set programming is able to search the holiday packages that best fits the customer needs.

The system has been developed under project “IDUM: Internet Diventa Umana” (project n. 70 POR Calabria 2000/2006 Mis. 3.16 Azione D Ricerca e Sviluppo nella Imprese Regionali - Modulo B Voucher Tecnologici) funded by the Calabrian Region. The project team involved five organizations: the Department of Mathematics of the University of Calabria (that has ASP as one of the principal research area), the consortium Spin, Exeura srl (a company working on knowledge management), Top Class srl (a travel agency), and ASPIdea (a software farm specialized in the development of web applications). The members exploited their specific knowledge for developing the innovative features of the system. The strong synergy among partners made possible to push the domain knowledge of the travel agency TopClass in both the ontology and in the reasoning modules. The result is a system that mimic the behavior of a seller of the agency and it is able to search in a huge database of automatically classified offers. IDUM combines the speed of computers with the knowledge of a travel agent for improving the efficiency of the selling process.

The IDUM system was initially conceived for solving the specific problems of a travel agency, and it is currently employed by one of the project partners: Top Class srl. Moreover, we received very positive feedbacks from the market, indeed many travel agents are willing to use the system, and the potential of IDUM has been recognized also by the chair of the Italian touring club, which is the most important Italian association of tour operators.

References

1. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. NGC **9** (1991) 365–385
2. Eiter, T., Gottlob, G., Mannila, H.: Disjunctive Datalog. ACM TODS **22**(3) (1997) 364–418
3. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV System for Knowledge Representation and Reasoning. ACM TOCL **7**(3) (2006) 499–562
4. Ricca, F., Gallucci, L., Schindlauer, R., Dell'Armi, T., Grasso, G., Leone, N.: OntoDLV: an ASP-based system for enterprise ontologies. Journal of Logic and Computation (2009)
5. Ricca, F., Leone, N.: Disjunctive Logic Programming with types and objects: The DLV$^+$ System. Journal of Applied Logics **5**(3) (2007) 545–573
6. Manna, M.: Semantic Information Extraction: Theory and Practice. PhD thesis, Dipartimento di Matematica, Universitá della Calabria, Rende, Cosenza Italia (2008)
7. Ruffolo, M., Manna, M.: HiLeX: A System for Semantic Information Extraction from Web Documents. In: ICEIS (Selected Papers). Volume 3 of LNBIP (2008) 194–209
8. Ruffolo, M., Leone, N., Manna, M., Saccà, D., Zavatto, A.: Exploiting ASP for Semantic Information Extraction. In: Proceedings ASP05 - Answer Set Programming: Advances in Theory and Implementation, Bath, UK (2005) 248–262
9. Gallucci, L., Ricca, F.: Visual Querying and Application Programming Interface for an ASP-based Ontology Language. In: Proceedings of the Workshop on Software Engineering for Answer Set Programming (SEA'07). (2007) 56–70
10. Gelfond, M., Leone, N.: Logic Programming and Knowledge Representation – the A-Prolog perspective . AI **138**(1–2) (2002) 3–38
11. Minker, J.: Overview of Disjunctive Logic Programming. AMAI **12** (1994) 1–24
12. Maedche, A., Staab, S.: Applying semantic web technologies for tourism information systems. In: Proc. of ENTER 2002, (2002)
13. Martin, H., Katharina, S., Daniel, B.: Towards the semantic web in e-tourism: can annotation do the trick? In: Proc. of the 14th ECIS 2006. (2006)
14. Jorge, C.: Combining the semantic web with dynamic packaging systems. In Proc. of AIKED'06 (2006), World Scientific and Engineering Academy and Society 133–138
15. Angel, G.C., Javier, C., Ismael, R., Myriam, M., Ricardo, C.P., Miguel, G.B.J.: Speta: Social pervasive e-tourism advisor. Telemat. Inf. **26**(3) (2009) 306–315
16. DERI: Digital Enterprise Research Institute. Technikerstrae 21a. Innsbruck, A.: `http://e-tourism.deri.at//`.
17. Terracina, G., Leone, N., Lio, V., Panetta, C.: Experimenting with recursive queries in database and logic programming systems. TPLP **8** (2008) 129–165