AI Magazine Volume 29 Number 2 (2008) (© AAAI)

Beyond the Elves: Making Intelligent Agents Intelligent

Craig A. Knoblock, José Luis Ambite, Mark Carman, Matthew Michelson, Pedro Szekely, and Rattapoom Tuchinda

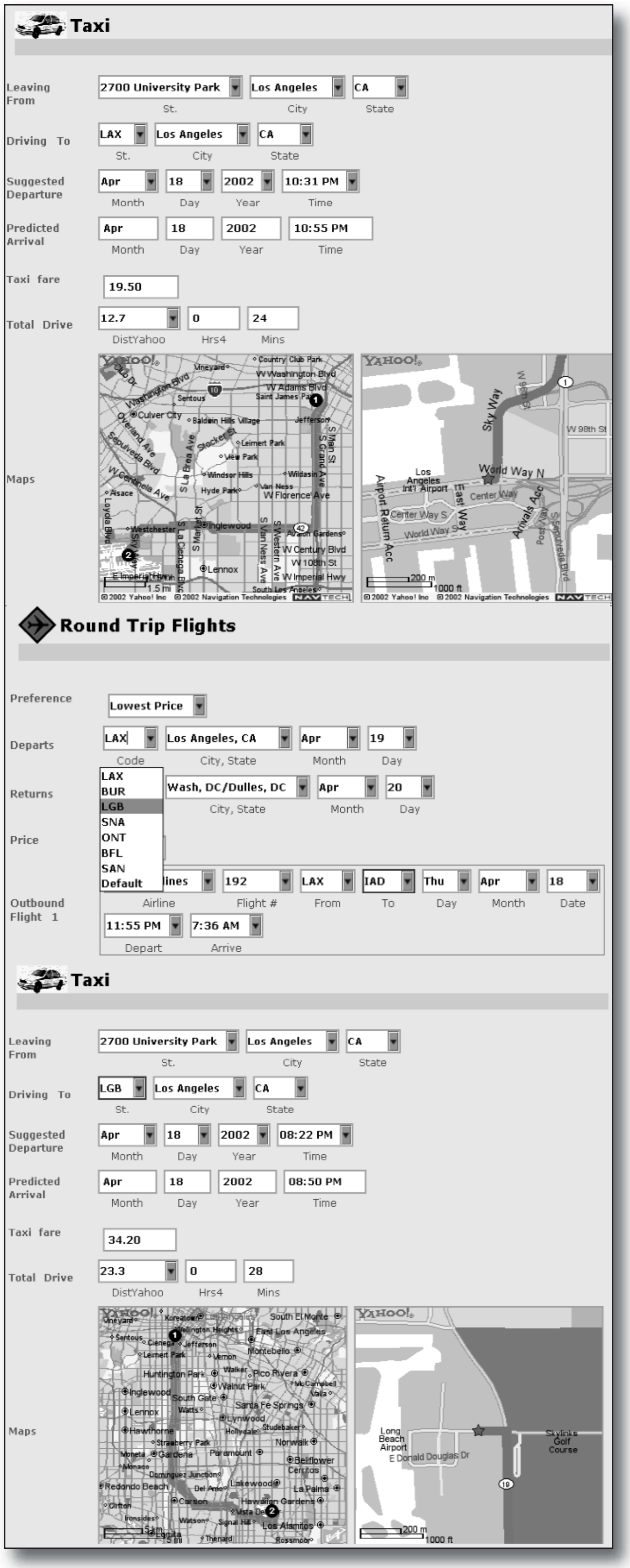
■ *The goal of the Electric Elves project was to develop software agent technology to support human organizations. We developed a variety of applications of the Elves, including scheduling visitors, managing a research group (the Office Elves), and monitoring travel (the Travel Elves). The Travel Elves were eventually deployed at DARPA, where things did not go exactly as planned. In this article, we describe some of the things that went wrong and then present some of the lessons learned and new research that arose from our experience in building the Travel Elves.*

The goal of the Electric Elves project (Chalupsky et al. 2001, 2002) at the University of Southern California (USC) was to develop software agents to support human organizations. The project was quite successful with impressive prototypes and many papers on the research. In fact, DARPA, which funded the project, was so enthusiastic about the results that it asked us to deploy a version of the Elves for use at DARPA. The original application of the Elves deployed at USC was for the office environment (called the Office Elves) (Scerri, Pynadath, and Tambe 2002; Pynadath and Tambe 2003) and required detailed information about the calendars of people using the system. Deploying this application at DARPA raised a host of issues including privacy and access to internal services. Thus, we decided to deploy a new application of the Electric Elves, called the Travel Elves. This application appeared to be ideal for wider deployment since it could be hosted entirely outside an organization and communication could be performed over wireless devices, such as cellular telephones.

The mission of the Travel Elves (Ambite et al. 2002, Knoblock 2004) was to facilitate planning a trip and to ensure that the resulting travel plan would execute smoothly. Initial deployment of the Travel Elves at DARPA went smoothly. Program managers and office directors began using the system. We trained the DARPA travel administrator on how to enter the travel information into the system. But over time things began to go wrong: agents would fail to correctly execute their assigned tasks, information got delayed or was not sent at all, the underlying sources were continually changing, and each user wanted the system to operate in different ways. In this paper, we first review the work on the Travel Elves and describe what the system did and how it worked. Then we describe the real-world problems that arose in deploying this application and the underlying causes of these failures. Finally, we will present some lessons learned and recent research that was motivated by our experiences in deploying the Elves at DARPA.

The Travel Elves

The Travel Elves introduced two major advantages over traditional approaches to travel planning. First, the Travel Elves provided an interactive approach to making travel plans in which all of the data required to make informed choices is available to the user. For example, when deciding whether to park at the airport or take a taxi, the system compares the cost of parking and the cost of a taxi given other selections, such as the airport, the specific parking lot, and the starting location of the traveler. Likewise, when the user is deciding which airport to fly into, the system not only provides the cost of the flights but also determines the cost of



Taxi

Leaving From	2700 University Park (St.)	Los Angeles (City)	CA (State)	
Driving To	LAX (St.)	Los Angeles (City)	CA (State)	
Suggested Departure	Apr (Month)	18 (Day)	2002 (Year)	10:31 PM (Time)
Predicted Arrival	Apr (Month)	18 (Day)	2002 (Year)	10:55 PM (Time)
Taxi fare	19.50			
Total Drive	12.7 (DistYahoo)	0 (Hrs4)	24 (Mins)	
Maps				

Round Trip Flights

Preference	Lowest Price			
Departs	LAX (Code)	Los Angeles, CA (City, State)	Apr (Month)	19 (Day)
Returns		Wash, DC/Dulles, DC (City, State)	Apr (Month)	20 (Day)
Price				

LAX
BUR
LGB
SNA
ONT
BFL
SAN
Default

Outbound Flight 1	lines (Airline)	192 (Flight #)	LAX (From)	IAD (To)	Thu (Day)	Apr (Month)	18 (Date)	
	11:55 PM (Depart)	7:36 AM (Arrive)						

Taxi

Leaving From	2700 University Park (St.)	Los Angeles (City)	CA (State)	
Driving To	LGB (St.)	Los Angeles (City)	CA (State)	
Suggested Departure	Apr (Month)	18 (Day)	2002 (Year)	08:22 PM (Time)
Predicted Arrival	Apr (Month)	18 (Day)	2002 (Year)	08:50 PM (Time)
Taxi fare	34.20			
Total Drive	23.3 (DistYahoo)	0 (Hrs4)	28 (Mins)	
Maps				

Figure 1. Heracles: Airport Propagates to Drive Subtemplate.

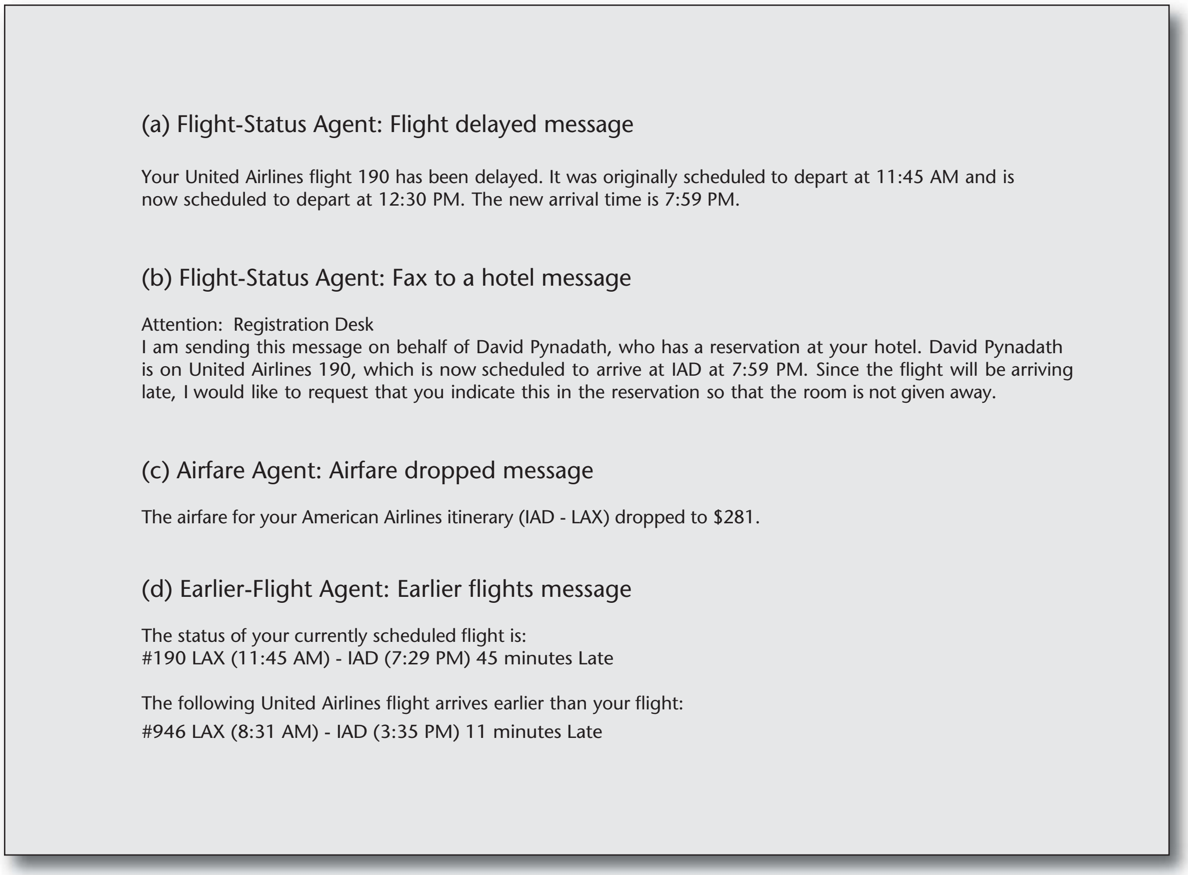
ground transportation at the destination, thus providing an accurate total cost. This cost information is integrated from several web sources.

Second, once a trip is planned, the Travel Elves launches a set of monitoring agents that attempt to make the trip as smooth and convenient as possible. The agents react to possible changes and respond according to the user preferences. For example, beyond simply notifying a traveler of flight delays, an agent will also send faxes to the hotel and car rental agencies to ensure that the room and car will be available.

These innovations in travel planning and monitoring are made possible by two underlying AI technologies. The first is the Heracles interactive constraint-based planner (Knoblock et al. 2001, Ambite et al. 2005). Heracles captures in a constraint network the relationships between all the information in the system, both retrieved from remote sources or resulting from user choices. The constraint network is partitioned into modules, called templates, that consist of closely related variables and constraints. The templates correspond to the task structure of the planning problem and are organized hierarchically similarly to a hierarchical task network. In addition, Heracles automatically generates a graphical user interface that is very intuitive for users. When the users make choices in the interface, the constraint network ensures that all the displayed data is consistent with the latest inputs. This happens behind the scenes, and users just experience that the information is aligned with their other choices.

The second innovation is the Theseus information agent platform (Barish and Knoblock 2005, 2002), which facilitates the rapid creation of information-gathering and monitoring agents with complex behaviors. These agents provide data to the Heracles planner and keep track of information changes relevant to the travel plans. Each agent corresponds to a streaming, dataflow Theseus plan. A plan may include data access operations (from web sources or databases), data manipulation (for example, relational algebra operations, aggregation), database updates, notifications operations (such as email or fax), as well as user-defined operations. The plans can combine the operations using conditionals and recursion. We briefly describe these technologies by example and invite the reader to consult the references for the details.

The Heracles user interface and the constraint network are illustrated in figure 1. Assume that the system has suggested that the user fly to Washington, D.C. departing from the Los Angeles International Airport (LAX) at 11:55 p.m. The user's preference is to arrive an hour before the departure time. Using a remote source (Yahoo! Maps) the system calculates that it takes 24 minutes to drive from the user's home to LAX, thus Heracles rec-


(a) Flight-Status Agent: Flight delayed message

Your United Airlines flight 190 has been delayed. It was originally scheduled to depart at 11:45 AM and is now scheduled to depart at 12:30 PM. The new arrival time is 7:59 PM.

(b) Flight-Status Agent: Fax to a hotel message

Attention: Registration Desk
I am sending this message on behalf of David Pynadath, who has a reservation at your hotel. David Pynadath is on United Airlines 190, which is now scheduled to arrive at IAD at 7:59 PM. Since the flight will be arriving late, I would like to request that you indicate this in the reservation so that the room is not given away.

(c) Airfare Agent: Airfare dropped message

The airfare for your American Airlines itinerary (IAD - LAX) dropped to $281.

(d) Earlier-Flight Agent: Earlier flights message

The status of your currently scheduled flight is:
#190 LAX (11:45 AM) - IAD (7:29 PM) 45 minutes Late

The following United Airlines flight arrives earlier than your flight:
#946 LAX (8:31 AM) - IAD (3:35 PM) 11 minutes Late

Figure 2. Actual Messages Sent by Monitoring Agents.

ommends leaving by 10:31 PM. When the user changes the departure airport from LAX to Long Beach airport (LGB), the system retrieves a new map and recomputes the driving time. Changing the departure airport also results in a different set of flights. The recommended flight from LGB departs at 9:50 p.m., and driving to LGB takes 28 minutes. Thus, to arrive at LGB by 8:50 p.m., the system now suggests leaving home by 8:22 p.m.

Given the final travel plan, Heracles automatically generates a set of Theseus agents for monitoring the plan. In figure 2 we show some of the messages sent by these agents. For example, the Flight-Status monitoring agent uses the ITN Flight Tracker site to retrieve the current status of a given flight. If the flight is on time, the agent sends the user a message to that effect two hours before departure. If the user's flight is delayed or canceled, it sends an alert through the user's preferred device (for example, a text message to a cellular telephone). If the flight is delayed by more than an hour, the agent sends a fax to the car rental counter to confirm the user's reservation. If the flight is going to arrive at the destination airport after 5 p.m., the agent sends another fax to the hotel so that the reserved room will not be given away.

Some of the other agents include the Airfare monitoring agent, which notifies the user if the price drops by more than the ticket rebooking fee; the Flight-Connection agent, which tracks the user's current flight and, a few minutes before it lands, sends the user the gate and status of the connecting flight; and the Restaurant-Finder agent that, on request, locates the user based on either a global-positioning system (GPS) device or his or her expected location according to the plan, and suggests the five closest restaurants, providing cuisine type, price, address, telephone number, latitude, longitude, and distance from the user's location.

What Went Wrong

The success of the Electric Elves project lead to a request from DARPA to deploy the system for use within DARPA. In early discussions with the DARPA technical staff it quickly became clear that this would be challenging for nontechnical reasons, including concerns about privacy and security as well as challenges in integrating with the DARPA internal systems. In response to these challenges, we decided to build the Travel Elves, which could operate on our own servers and would be largely independent of the internal DARPA systems. Since DARPA already has a travel office that handles travel plans and reservations, we decided to deploy just the portion of the system for monitoring travel. The one challenge was how to get the detailed information about a user's travel plans into the system. To address that problem, we built a simple interface that made it possible to quickly enter the detailed travel plans.

The initial testing and deployment went well. We starting using the system ourselves and discovered we often had much better access to the relevant data than even the organizations that controlled the data. For example, landing at an airport, the airline might announce that it had no connecting flight information available, but we found the arrival gate, departure gate, and departure time of the connecting flight would all be available from the Elves. Initial testing and use looked very promising for a compelling demonstration of the Elves technology.

The DARPA travel office began entering real travel plans, and a DARPA office director, a deputy director, and several program managers began using the system. Unfortunately, problems began to arise. There were a number of causes of these problems. First, the most common source of problems was failures in the data sources. Sources went down, sources changed, or they simply generated unexpected results. A challenge with the Elves was that we depended on many different sources of information that we had little control over, and if any of them changed or became unavailable, then problems could arise. In addition, some of the data sources were simply unreliable. For example, in one instance, the DARPA program manager contacted us to find out why the Elves had sent the reminder message about his connecting flight after the flight had already departed. It turns out that this was caused be a time zone exception in a web source where the particular airport he had been traveling through was in a different time zone than what the Elves had thought, so the message got sent an hour late.

A second significant source of problems was that it was very difficult to determine in advance exactly what we wanted the Elves to do in various situations. The best example of this was that we wanted the system to notify a user of delays and cancellations when they happened so that a traveler would have as much time as possible to deal with the problem. Naturally, a flight delay then occurred in the middle of the night, resulting in a message to a user's cell telephone at 3 a.m. to say that the 8 a.m. flight had been delayed. In this case the agent worked as designed, but not everyone wants a message at 3 a.m. about a flight delay. This is an example of the larger issue that came up repeatedly, which was that each user wants the agents to behave in slightly different ways depending on how they use the system. Ideally, we would like a system where the users could individually define their own monitoring agents.

A third source of problems was that the system operated as planned but produced unexpected behaviors. A good example of this is the monitoring agent that sent messages whenever the ticketed price changed. The purpose of this agent was to provide opportunities for price saving when a ticket price dropped, but early on we did not distinguish price decreases from increases. We were also surprised by the number and frequency of the price changes, many of which could be quite small, and they would sometimes get annoying. Later we distinguished price decreases from increases and set a threshold for a total change before a user would be notified of the change (it had to be more than the change fee for it to make sense to notify the user).

Lessons Learned

In the remainder of this article we describe some of the technologies that arose due to our experience from the Elves and the challenges in dealing with many of the real-world data sources. In particular, we describe our work on automatically modeling, integrating, and monitoring new sources of data, methods for allowing end users to define their own agents tailored to their individual needs, and finally, work on trying to exploit some of the unexpected behaviors, such as the frequent price changes.

Discovering New Sources

A key aspect of Travel Elves was its ability to monitor Web sources and provide relevant information to users. A problem it faced was that those sources could sporadically become unavailable. One solution is to program the agents to access multiple redundant sources. Augmenting an existing system with new sources can be very time consuming, however, and a more sustainable approach is to have the agents discover those sources themselves. Unfortunately, since service providers rarely agree on common service interfaces (aka parameter names and method signatures), the problem of service discovery is nontrivial. It can be divided into three phases as shown in figure 3.

The first phase involves searching for relevant sources by accessing a service registry (for example, UDDI) or Web index (such as Google). In our example, the keywords "airport weather" are used to discover a service named NOAAWeather. To improve search performance, systems have been developed for classifying services into predetermined categories (Heß and Kushmerick 2003) and

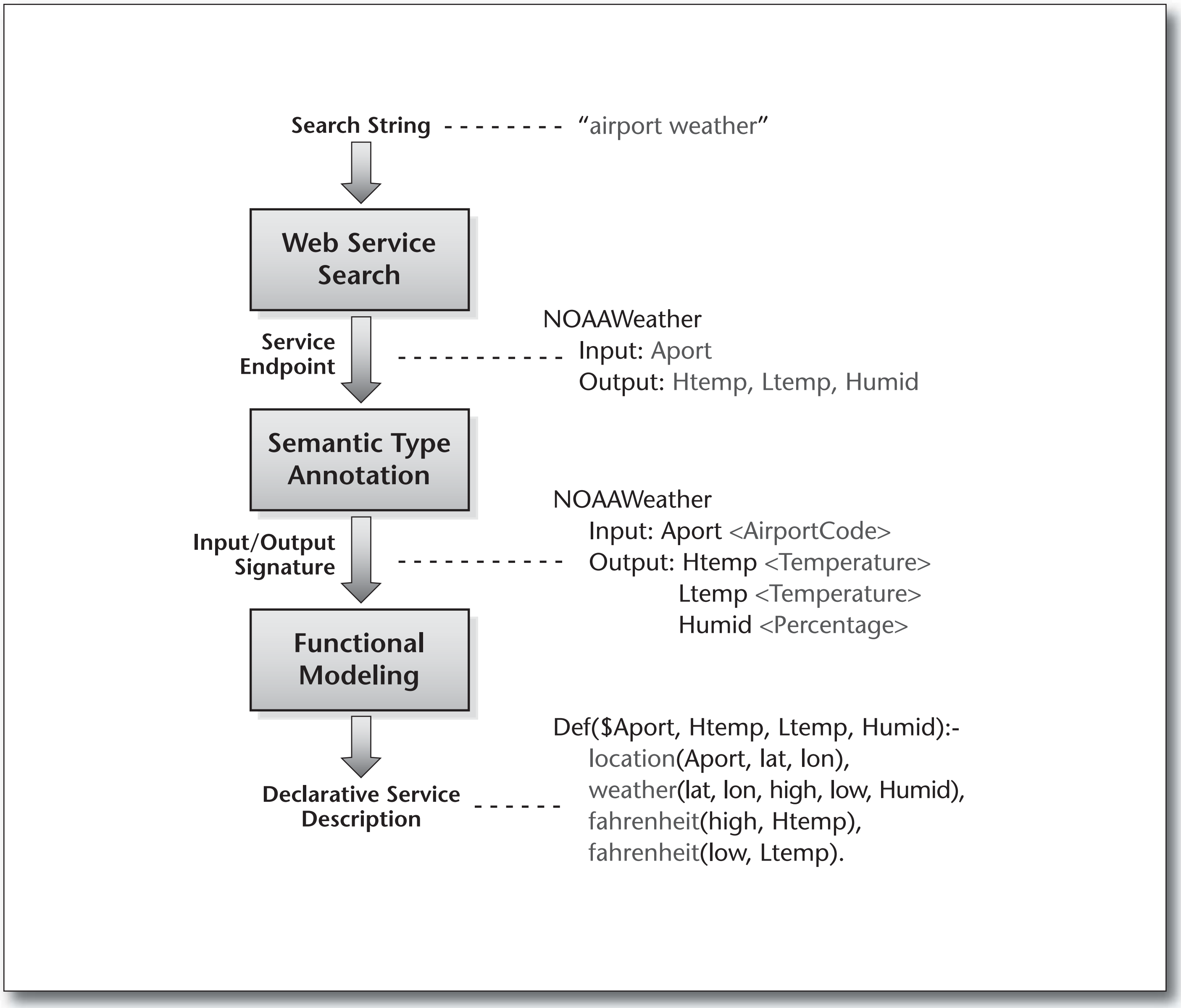
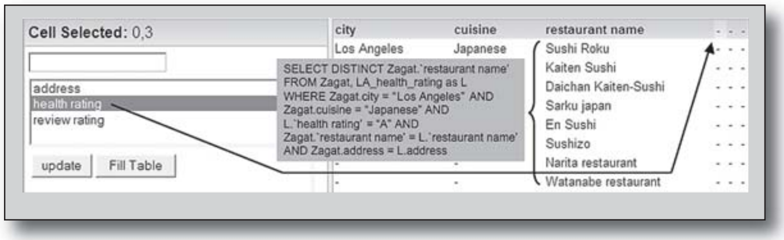


Figure 3. Discovering and Modeling New Information Sources.

clustering similar services together (Dong et al. 2004).

In the second phase, we start to model what the service does by determining what type of data it requires as input and produces as output. We assign semantic types such as Temperature (as opposed to syntactic types like integer) to the attributes of the service. This task can be viewed as a classification problem where the metadata labels (for example, "Htemp") are the features (Heß and Kushmerick 2003). In our work (Lerman, Plangrasopchok, and Knoblock 2006), we have extended the set of features to also include the data generated by the service (for example, "21?F"), resulting in improved performance.

In the third phase, we determine how the output is related to the input. In the example, the service returns two temperature values. In order to use the service, we will need to know which if either of these is the high temperature. This information is described in the source definition at the bottom of the figure. We have built a system capable of inducing such definitions automatically (Carman and Knoblock 2007b, Carman 2006, Carman and Knoblock 2007a). The system works by searching the space of plausible source definitions and comparing the output they produce with that of the new source.



Cell Selected: 0,3

city	cuisine	restaurant name
Los Angeles	Japanese	Sushi Roku
		Kaiten Sushi
		Daichan Kaiten-Sushi
		Sarku japan
		En Sushi
		Sushizo
-	-	Narita restaurant
-	-	Watanabe restaurant

```
SELECT DISTINCT Zagat.'restaurant name'
FROM Zagat, LA_health_rating as L
WHERE Zagat.city = "Los Angeles" AND
Zagat.cuisine = "Japanese" AND
L.'health rating' = "A" AND
Zagat.'restaurant name' = L.'restaurant name'
AND Zagat.address = L.address
```

Figure 4. Example of a Simple Table, and the Query That Karma Induces Based on the Data Examples the User Has Provided.

Building Agents without Programming

The Travel Elves are personal agents working on behalf of individual users. Because all users have slightly different requirements and preferences, a one-size-fits-all agent is not appropriate. Users of the Travel Elves reported that they wanted the agents to adapt or wanted to be able to easily customize their agents. The main impediment for building or customizing agents is programming. Over the past few years, we have worked on tools to enable casual users to create agents without having to program. These agents' behavior can range from simple (that is, retrieving data from a web) to complex (that is, monitoring data from multiple web sources and acting upon it once the data satisfies the criteria specified by a user.)

Initially, we developed a tool called Agent Wizard (Tuchinda and Knoblock 2004) that let users construct Travel Elves agents by answering questions. The agent-building process is similar to online tax software, where future questions are based on what users answer in the past. However, the limitation of Agent Wizard is that when there are many data sources integrated into an agent, users have to answer too many questions on how to integrate those data sources together. We address this limitation in our current tool called Karma (Tuchinda, Szekely, and Knoblock 2007). In Karma, users provide examples of data they want, and the system automatically constructs a plan that delivers the requested data by integrating multiple sources. Karma starts by presenting the user a blank table into which users can type examples of data. Initially, when the table is empty, there are no constraints, but as users enter values, constraints are formulated to select relevant data sources and attributes available for users to select from in each column of the table. When users select a blank cell, a menu allows the user to type partial values. Karma then offers suggestions for completing the value typed so far by matching against all values in all sources that are consistent with the value the user has typed so far. Once users specify a few examples, the system offers options for filling other cells in the table and can also make suggestions about additional columns of data that are compatible with the information users have entered so far. When the constraints are satisfied uniquely, Karma automatically fills the relevant blank cells.

For example, figure 4 shows a snapshot of Karma where the user's goal is to retrieve reviews of Japanese restaurants in Los Angeles and their corresponding health rating. In this example, the restaurant review information and the health rating information belong to different sources. The user starts from the blank table by providing example values of "Los Angeles" and "Japanese." Karma automatically deduces the corresponding attribute and fill outs attributes city and cuisine. Next, the user selects the attribute restaurant name from the suggestion list on the left side. At any point in time, the user can click the fill button and Karma selects the relevant sources and automatically constructs the join conditions that integrate the data from these sources in a way that satisfies the constraints. The orange box in the figure shows the queries that Karma produces based on the constraints it has induced from the examples and the source definitions to populate the restaurant list. The key feature of Karma is that the user never has to see those queries, which can grow quite complex for larger tables.

Predicting Prices

One of the issues that arose during the deployment of Travel Elves was the excessive number of notifications about airfare changes. The price of a flight can change multiple times in a day. So, even with travel plans laid out, the traveler is often left with the question of when to buy the tickets.

To address this issue we decided to study ticket pricing, so we monitored the pricing of air tickets of two direct domestic routes over the course of two months. Initially, the pricing appeared to be too haphazard to see any pattern. Figure 5 shows the very erratic pricing of a single flight collected over a one-month period. However, the airlines are using an underlying revenue management algorithm that adjusts the price according to some set of criteria that includes factors such as the seat availability, number of days before take off, and so on.

Since airlines fly the same fixed schedule day after day or week after week, we realized that we could monitor a particular flight from one day to the next to capture the overall pricing behavior for that daily or weekly flight. For example, we could gather the airfare for every American Airlines flight 192 that departs from Los Angeles International airport (LAX), which would be roughly 30 flights a month. By exploiting the pricing of the flights on all of the previous days, we realized that we could then predict the pricing of the flight on same day in the future. Thus, we implemented a system called Hamlet[1] (Etzioni et al. 2003), which learns the hidden pricing models implemented by the

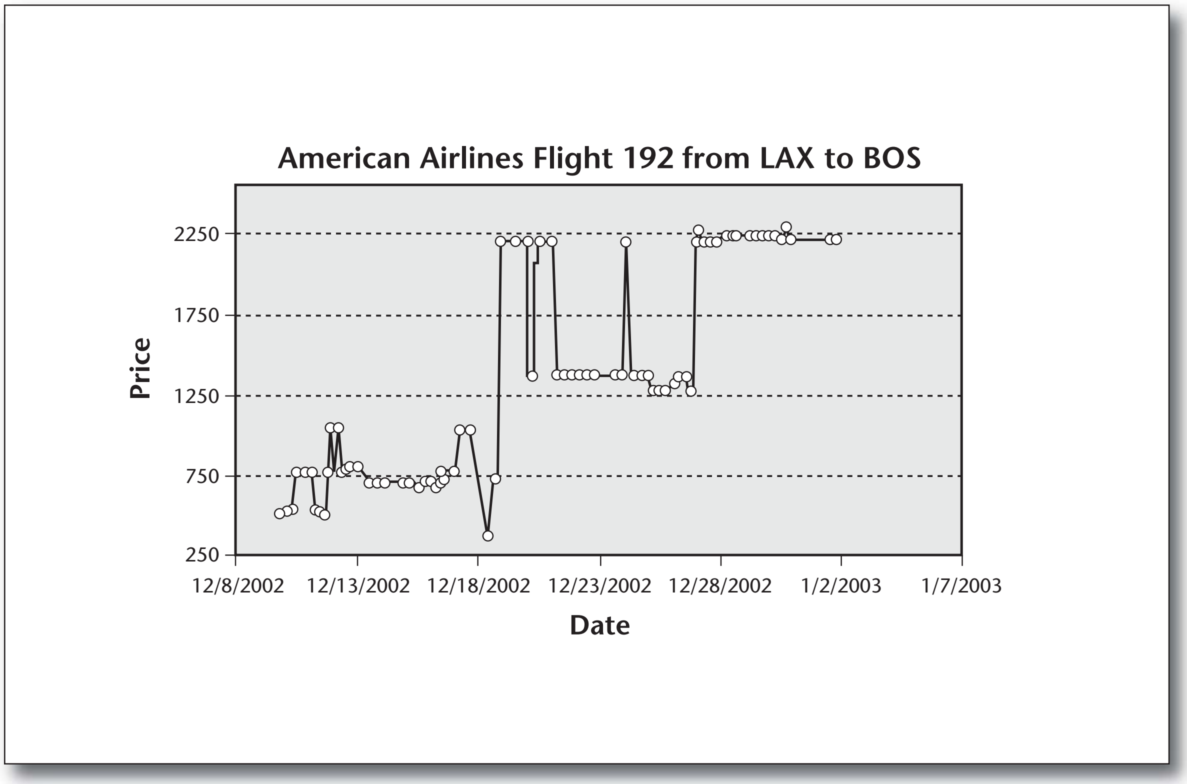


Figure 5. Pricing for American Airlines Flight 192 Departing LAX on January 2, 2004, and returning from BOS on January 9, 2004.

airlines. Hamlet works by combining the results of three learning algorithms through stacked generalization (Ting and Witten 1999): time series (Granger 1989), Q-learning (Sutton and Barto 1998), and rule learning (Cohen 1995). Hamlet can predict with surprisingly high accuracy whether a traveler should buy a ticket right away or wait to buy a ticket to minimize the cost of a trip. For example, in a simulation using real-world flight data, Hamlet was able to achieve a savings of 61.8% of the optimal possible savings. The technology underlying Hamlet was licensed to a company called Farecast (www.farecast.com), and this company now collects data on routes throughout the United States, and travelers can go to this site to get predictions to answer the question of when to buy an airline ticket.

Discussion

In this article we have described specific research directions that we have pursued to address some of the issues that arose in the Elves. More generally, there is still the need to develop general solutions to building agents that can operate in the real world. In particular, we need agents that can robustly accomplish their tasks, responding appropriately to failure, that can learn from their past experience, that can communicate flexibly with humans and other software agents, and that can explain their behavior both on success and failure. DARPA has continued to fund work on this topic in the DARPA personalized agents that learn (PAL) program, where the focus is on developing learning technology so that agents can learn to solve tasks "in the wild." While we have made significant progress on many of these issues, there is still much work to be done before we will consider these agents intelligent.

Acknowledgements

This research is based upon work supported in part by the National Science Foundation under Award No. IIS-0324955, in part by the Air Force Office of Scientific Research under grant number FA9550-04-1-0105, and in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010. The U.S. government is authorized to reproduce and distribute reports for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

Three Anecdotes from the DARPA Autonomous Land Vehicle Project

Dan Shapiro

The DARPA Autonomous Land Vehicle (ALV) was a 12-foot tall, eight-wheeled robot with multiple sensors, tasked to go from point A to point B without human intervention in the hills outside of Denver in about 1985. This was a large applied research effort that presented many opportunities for unusual experiences.

In one such experience, I was called in, at the last minute, to help improve our ALV proposal. The proposal was a 300-page document that segued smoothly from problem description to corporate capabilities and managerial plan, omitting any mention of technical approach. This taught me a rule of thumb I have seen validated many times: the larger the project (in dollars and scope), the poorer the technical proposal.

In a second experience, I was demonstrating a dynamic programming algorithm at a quarterly review. The algorithm found minimum time paths from all points to a goal, set in an open field. I clicked a starting point, and the system traced a route that detoured to use a nearby road. I clicked closer to the goal, and the detour was more pronounced. When I clicked right next to the goal, the optimal path was horribly convoluted—it went away from the goal and to a road, which it followed around a long hogback ridge before departing cross-country for the objective. I had to offer the sponsor something by way of explanation, so I looked at him and said, “We just spent $200,000 of your money to learn that roads are a good idea.”

A third experience occured when I was at the computer, writing code for a new route-planning algorithm. A senior manager and a vice president of the firm came in, so I told them what I was doing. One said, “make it more AI-like,” while the other wanted “a more engineering approach.” That goal conflict literally ran backwards from my fingertips, through my corporate management, and into DARPA, where they were debating the same question. The resolution pushed the ALV project in the applied direction and influenced later DARPA efforts.

Daniel Shapiro (dgs at stanford.edu) is the executive director of the Institute for the Study of Learning and Expertise (ISLE), and president of Applied Reactivity, Inc. (ARi).

Note

1. “To buy or not to buy....”

References

Ambite, J. L.; Barish, G.; Knoblock, C. A.; Muslea, M.; Oh, J.; and Minton, S. 2002. Getting from Here to There: Interactive Planning and Agent Execution for Optimizing Travel. In *Proceedings of the Fourteenth Conference on Innovative Applications of Artificial Intelligence* (IAAI-2002), 862–869. Menlo Park, CA: AAAI Press.

Ambite, J. L.; Knoblock, C. A.; Muslea, M.; and Minton, S. 2005. Heracles II: Conditional Constraint Networks for Interleaved Planning and Information Gathering. *IEEE Intelligent Systems* 20(2): 25–33.

Barish, G., and Knoblock, C. A. 2002. Speculative Execution for Information Gathering Plans. In *Proceedings of the Sixth International Conference on Artificial Intelligence Plan-*

Simplicity Rather Than Knowledge

William Bricken

I've spent over 20 years implementing algorithms for Boolean minimization, with particular application to area optimization of large semiconductor circuits. This exploration has included almost all published algorithms, dozens of conventional and exotic data structures, over two dozen programming languages and styles, and almost every level of implementation, from reconfigurable silicon through customized instruction sets, conventional programming languages, and very high-level languages such as Mathematica.

The huge, randomly generated CNF formulae used to study SAT phase transition have attracted many creative approaches (such as variants of unit propagation, differential equations, probabilistic moments, component connectivity, cutting planes, and so on). However, I've learned one thing about the nature of Boolean minimization that seems obvious now. No matter how clever an algorithm is, no matter how extensively the structure of a problem is analyzed, no matter how much adaptive learning and lemma caching is used, the most successful approach to the general Boolean minimization problem is to use the simplest possible algorithm—brute force applied to lexicographically sorted formulae and implemented in reconfigurable silicon.

The lessons learned? For the general case, complexity cannot be finessed by any degree of cleverness. Native silicon is faster than any symbolic optimization. Simplicity rather than knowledge is the key.

William Bricken teaches mathematics at Lake Washington Technical College, while working on axiomatic foundations incorporating spatial representation of logic and integers.

ning and Scheduling (AIPS 2002), 184–193. Menlo Park, CA: AAAI Press.

Barish, G., and Knoblock, C. A. 2005. An Expressive Language and Efficient Execution System for Software Agents. *Journal of Artificial Intelligence Research* 23: 625–666.

Carman, M. J. 2006. Learning Semantic Definitions of Information Sources on the Internet. Ph.D. Dissertation, Department of Information and Communication Technologies, University of Trento, Trento, Italy.

Carman, M., and Knoblock, C. A. 2007a. Learning Semantic Definitions of Online Information Sources. *Journal of Artificial Intelligence Research* 30: 1–50.

Carman, M. J., and Knoblock, C. A. 2007b. Learning Semantic Descriptions of Web Information Sources. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence* (IJCAI-07). Menlo Park, CA: AAAI Press.

Chalupsky, H.; Gil, Y.; Knoblock, C. A.; Lerman, K.; Oh, J.; Pynadath, D. V.; Russ, T. A.; and Tambe, M. 2001. Electric Elves: Applying Agent Technology to Support Human Organizations. In *Proceedings of the Conference on Innovative Applications of Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Chalupsky, H.; Gil, Y.; Knoblock, C. A.; Lerman, K.; Oh, J.; Pynadath, D. V.; Russ, T. A.; and Tambe, M. 2002. Electric Elves: Agent Technology for Supporting Human Organizations. *AI Magazine* 23(2): 11–24.

Cohen, W. W. 1995. Fast Effective Rule Induction. In *Proceedings of the Twelfth International Conference on Machine Learning,* Tahoe City, California, 115–123. San Francisco: Morgan Kaufmann Publishers.

Dong, X.; Halevy, A. Y.; Madhavan, J.; Nemes, E.; and Zhang, J. 2004. Similarity Search for Web Services. In *Proceedings of the Thirtieth Very Large Data Bases Conference* (VLDB), Toronto, Ontario. San Francisco: Morgan Kaufmann Publishers.

Etzioni, O.; Knoblock, C. A.; Tuchinda, R.; and Yates, A. 2003. To Buy or Not to Buy: Mining Airline Fare Data to Minimize Ticket Purchase Price. In *Proceeding of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: Association for Computing Machinery.

Granger, C. W. J. 1989. *Forecasting in Business and Economics,* 2nd ed. New York: Harcourt Brace.

Heß, A., and Kushmerick, N. 2003. Learning to Attach Semantic Metadata to Web Services. In *Proceedings of the Second International Semantic Web Conference* (ISWC), Sanibel Island, Florida. Lecture Notes in Computer Science 2870. Berlin: Springer.

Knoblock, C. A. 2004. Building Software Agents for Planning, Monitoring, and Optimizing Travel. In *Information and Communication Technologies in Tourism 2004: Proceedings of the Eleventh International Conference on Information Technology and Travel and Tourism,* 1–15. Berlin: Springer.

Knoblock, C. A.; Minton, S.; Ambite, J. L.; Muslea, M.; Oh, J.; and Frank, M. 2001. Mixed-Initiative, Multisource Information Assistants. In *Proceedings of the Tenth World Wide Web Conference*, Hong Kong, 697–707. New York: Association for Computing Machinery.

Lerman, K.; Plangrasopchok, A.; and Knoblock, C. A. 2006. Automatically Labeling the Inputs and Outputs of Web Services. In *Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence* (AAAI-06). Menlo Park, CA: AAAI Press.

Pynadath, D. V., and Tambe, M. 2003. An Automated Teamwork Infrastructure for Heterogeneous Software Agents and Humans. *Journal of Autonomous Agents and Multi-Agent Systems* 7(1–2): 71–100.

Scerri, P.; Pynadath, D. V.; and Tambe, M. 2002. Towards Adjustable Autonomy for the Real World. *Journal of Artificial Intelligence Research* 17: 171–228.

Sutton, R. S., and Barto, A. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: The MIT Press.

Ting, K. M., and Witten, I. H. 1999. Issues in Stacked Generalization. *Journal of Artificial Intelligence Research* 10: 271–289.

Tuchinda, R., and Knoblock, C. A. 2004. Agent Wizard: Building Information Agents by Answering Questions. In *Proceedings of the 2004 International Conference on Intelligent User Interfaces*, Funchal, Madeira, Portugal. New York: Association for Computing Machinery.

Tuchinda, R.; Szekely, P.; and Knoblock, C. A. 2007. Building Data Integration Queries by Demonstration. In *Proceedings of the 2007 International Conference on Intelligent User Interfaces*, Honolulu, Hawaii. New York: Association for Computing Machinery.

Craig A. Knoblock is a senior project leader at the Information Sciences Institute and a research professor in computer science at the University of Southern California (USC). He is also the chief scientist for both Fetch Technologies and Geosemble Technologies, which are spinoff companies from USC. He received his Ph.D. in computer science from Carnegie Mellon in 1991. His current research interests include information integration, automated planning, machine learning, and constraint reasoning and the application of these techniques to geospatial data integration.

José Luis Ambite is a research assistant professor at the Computer Science Department and a senior research scientist at the Information Sciences Institute and at the Digital Government Research Center, all at the University of Southern California. His research interests include information integration, databases, knowledge representation, AI planning, and digital government. His current focus is on web service composition and in efficiently creating and querying large entity bases. He received his Ph.D. in computer science from the University of Southern California in 1998.

Mark Carman is a postdoctoral researcher at the Informatics Faculty of the University of Lugano, Switzerland. He obtained a Ph.D. from the University of Trento, Italy, while working at the Information Sciences Institute of the University of Southern California. His work spans research areas in AI planning, data Integration, and information retrieval.

Matthew Michelson is a Ph.D. student at the University of Southern California. His research interests include information integration, information extraction, and record linkage. He received his BS from the Johns Hopkins University in 2002 and his MS from USC in 2005.

Pedro Szekely is a research assistant professor in the Computer Science Department of the University of Southern California. He is also a project leader and a research scientist at USC's Information Sciences Institute. His research interests span multiagent systems, planning, scheduling, decision support, human computer interaction, and visualization. He has published numerous papers in journals and conferences and has held senior program committee positions, such as conference chair of IUI 2000. Szekely received his Ph.D. in computer science from Carnegie Mellon University in 1987.

Rattapoom Tuchindais is a Ph.D. student at USC and consultant for Farecast, Inc., a company based on his research work on airfare prediction. His research interests include information integration, user interface, agent execution, mixed-initiative planning, and machine learning. He received a BS in electrical engineering and computer science and an MS in engineering from the Massachusetts Institute of Technology. He can be reached at USC/Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292; pipet@isi.edu, www.isi.edu/pipet.