

A Mobile Sales Force Automation System based on an Agent-Oriented Natural Language Interface

Babak Hodjat Ph.D.¹, Cristobal Baray Ph.D.², Julian Tandler³

Problems with Current Mobile Sales Force Automation Solutions

Industry experts estimate that many mobile Sales Force Automation (SFA) applications fail to deliver a real return on investment. These estimates are as high as 50% to 80% [1]. Why? A major reason is lack of usage. Current wireless access solutions force users to navigate complex application menus and memorize difficult commands. Combine these limitations with the small screens on wireless devices and spotty network coverage and you have a recipe for failure. If the wireless solution is cumbersome and painful to use, the mobile sales professional simply won't waste their time with it.

Today's wireless networks are unreliable, resulting in incomplete coverage, high latency, and frequent disconnects. These limitations have not been considered in the design and development of current wireless SFA solutions. More often than not, they have menu heavy interfaces that require users to navigate through numerous screens to get to the desired information or data entry form. This can cause major headaches, when on average it takes up to 60 seconds for a screen to load onto a wireless web browser. In addition, querying capabilities are often limited to small sets of pre-defined reports created by the user or system administrator while in the office. This is due to the difficulty of porting complex reporting wizards onto tiny wireless web interfaces. If ad hoc querying capabilities are available, they often require the user to adhere to strict protocol, making it nearly impossible to find exactly what they are looking for.

Natural Language Interfaces to Mobile Applications

When deploying a wireless SFA solution, the objective should be to provide a tool that enables the user to focus on the customer, rather than worrying about how to access and update the information. Empowering users with access to their sales data via a Natural Language Interface (NLI) [2] does just that. Rather than taking the time to learn how to navigate through extensive menus only to be forced to sit through multiple screen refreshes, users simply ask for what they want in one step. For example, a salesperson wanting to access their company's funnel information for the Western region and for deals larger than \$1 million can request "Show all Western regional sales opportunities greater than \$1 million," or just "all West opps > \$1m", and view the specific report delivered in real-time on their wireless device. In other words, with NLI, the user's task can change from expressing a series of procedural steps to the system to expressing the goal to the system. It is our view, that this expression of the goal is a more appropriate and intuitive for interaction between humans as well as between human and machine.

¹ Dejima Inc., 160 W Santa Clara St., San Jose, CA 95113, Babak.hodjat@dejima.com

² Dejima Inc., 160 W Santa Clara St., San Jose, CA 95113, cbaray@dejima.com

³ Dejima Inc., 160 W Santa Clara St., San Jose, CA 95113, julian.tandler@dejima.com

With a NLI, SFA vendors can provide their customers with access to sales data via multiple interfaces such as email or the wireless web. Using wired or wireless email, users simply send an email to a select address [ex: salesforce@ABC_company.com] with their request for action. The NLI interprets their request, executes the desired action in the target application, and delivers the result in a reply to their initial email. A wireless web interface augmented with a NLI provides users with ultimate flexibility, as they can use a combination of free-form text requests and a graphical user interface (GUI). Users simply ask for the information they need in their own language or when convenient they point and click their way to the desired data or transaction. Users are allowed to use the most appropriate modality or combination of modalities for the particular task at hand.

Providing access to sales data via a Natural Language Interface is a cost effective, flexible solution that overcomes the limitations of wireless devices and networks, boosting usage of mobile CRM applications.

The most important drawback in using NLIs is managing the user's expectations of the system functionality. Unlike GUIs, NLIs do not provide a defined scope of the end application, while at the same time giving the impression that the system can understand, and therefore do, anything that the user requests. To this end, extra care should be taken in how the NLI is designed and presented to users, enabling the users to get a sense of the limitations of the functional scope through interactions with the NLI. This is all the more important in a mobile setting, where the NLI does not have the luxury of a complementing GUI.

An Agent-Oriented NLI

The Adaptive Agent Oriented Software Architecture (AAOSA) [3] is an Agent Oriented Software Engineering (AOSE) [4] system used for Natural Language Interfaces (NLI) [5]. To create an NLI using AAOSA, an engineer will model the application, not the language, in the form of an agent network. Each agent is responsible for a semantic concept in the application, and reports to an agent up-chain, which represents a broader concept. When an agent network is presented with an input request, agents make claims to a role in responding to the input. The strength of the claims in agents varies depending on the criteria on which the claims are based: patterns in the request, dialog history, application context, etc. This claim-making behavior of agents is defined in each agent using rules called policies (see figure 1).

Policies are the only language dependent elements in an AAOSA agent network, making the porting of such networks to other languages a fraction of the development time itself. For example porting an English SFA agent network to Japanese was accomplished in a matter of weeks.

AAOSA also allows for dynamic addition of policies and agents to an agent network, making it suitable for the heavy customization requirements of enterprise applications with a single reference agent network core. This is especially useful for large enterprise deployments or application service provider (ASP) model CRM solutions.

AAOSA, being a software engineering tool, allows for reusability of agent sub-networks, enabling enterprise applications to rapidly tailor and customize new business objects and process flows as their enterprise applications evolve.

The deployed application

Our application has been deployed at Salesforce.com as their Wireless Edition (see figure 2). Regardless of what modality or device

the Salesforce.com user prefers, our application takes their query, makes transactions against Salesforce.com on behalf of the user, and presents the user with the results of their query, in an intuitive, useful manner. Similar to Salesforce.com's web-based application, the Wireless Edition is deployed on the server side providing a device agnostic service with no client software requirements (see figure 3).

Application architecture

The components in our application break down into three basic sets. Interaction components communicate with the end user, actuation components communicate with Salesforce.com, and the Dejima AAOSA agent network maps the user's natural language request into a set of transactions against Salesforce.com (see figure 4).

The interaction components are responsible for both gathering information from the user and presenting results back to the user. These are written for specific devices and communication protocols (email, http, wap, etc). The separation of the presentation layer, similar in spirit to the model-view-controller design pattern, gives us the benefit of using the same agent solution across all modalities. Each interaction component obtains the user's input via a method suited for the particular protocol (subject line of an e-mail, an input box over the web), passing the input along to our agent network. Once a result is generated by the actuation components, each interaction component will render the data in a way appropriate for the user's modality. Reports in plain text are created as replies to email requests. Small, efficient web tables are created for the wireless web interface, with the appropriate links.

The actuation components represent our interface into the Salesforce.com application. Our application abstracts the basic salesforce automation application functions and subclasses from that framework into a specific solution to integrate with Salesforce.com. Salesforce.com provides an XML-RPC interface, which provides access to their core functionality: validating login credentials, reporting (data and meta-data), and updating/creating user records. Our classes use that XML-RPC interface to carry out the actions defined in the user's natural language request.

Calculating the right set of actions, given a user's request is not a trivial task. End users' requests are domain, company, and user specific. Of course, their requests are focused on a common workflow centered on Salesforce.com. But each Salesforce.com customer also has the opportunity to customize the application for their own needs; adding fields to records that are important to their own industry/workflow/process. Finally, each user will have access to a different set of records, fostering very different requests across the user base. Our agent network handles each of these issues in a slightly different way.

The domain knowledge is represented by the overall architecture of the agent network. Each object in Salesforce.com is its own separate subdomain - a sub-network is responsible for deciding when the user is requesting information about accounts, likewise, another sub-network manages opportunity requests, etc. These sub-networks fall underneath agents responsible for deciding what action the user would like to take. If one is requesting a report on a specific contact or if one is updating contact information, the same sub-network is used to identify contact information, but different agents recognize the update versus the report command.

Company knowledge, such as custom fields for each object are handled by specific agents with each sub-network. We take advantage of the meta-data provided by Salesforce.com's XML-RPC API. From the meta-data, we get a description of each field within each object in the company. The description describes the name, label, type and other attributes about each field. Most of that is useful for presentation and validation of the input. For interpreting the user's input, we focus on the label of the field (sometimes the type is useful). These custom field agents scan the user's input for references to custom fields (by label names, and when appropriate, custom pick-list values). If these agents do recognize a field reference, they make the claim, allowing the sub-network to signal the user's intent. This mechanism customizes the agent network for each company, without any effort on the end user's part.

User knowledge is handled in a slightly different way. User requests based on their own information is much more relevant to the customer, but more difficult to handle on the application's side. A request like "opportunities closing this month" is independent of the user's data - the report would be the same for John Doe and Jane Doe. But when John requests for his contacts at Company X, he's using a vocabulary that might not be the same as Jane's. Thus, our interpretation engine must adapt to each user's particular database. To do this, the user's request is passed through various filters, trying to identify which tokens in the user's input refer to actual data within the user's database. If the user's input is "accounts in san jose" - we determine before any interpretation has begun that san jose is in the user's database as a city. Or if the user's input was "opportunities at San Jose Industrial", we'd also determine beforehand that San Jose Industrial is a company name in the user's database. Combined with our agent network, we're able to interpret user's requests, including the phrases specific to the current data.

Usage

Since deployment, the AAOSA-NLI for Sales Force Automation has enjoyed a strong uptake. In the first 120 days since the announcement of the service more than 300 companies signed up for use of the system. Usage has been evenly divided between the e-mail and wireless web modalities. The success-rate of the system has been consistently above 90%, and more than 95% of the queries have been within the functional scope of the system capabilities. The average use per user of the system in the first few weeks of deployment was around 5.2 hits per week. Trends show a steady growth in usage and subscriptions since deployment.

Related Methods

Currently, the most popular approach to the problem noted above is through synchronization solutions. These solutions suffer greatly from the complexity of the interface, and normally require much processing power and capacity on the handheld devices.

Speech recognition based systems, such as systems based on VoiceXML, can provide mobile access to applications similar to the SFA domain. Although DDSFA can also be voice enabled, voice is not always the preferred modality of access, and current voice based systems do not provide as natural and free form an interface as to be effective in addressing the entire application space, in a useable fashion.

Conclusion

Shrinking desktop applications to fit mobile devices is not the solution to enabling mobile access to enterprise applications. Limited screen space, connectivity issues, and input modality restrictions in present-day mobile devices call for creative usable interaction solutions such as that provided by AAOSA-NLI. Our experience in creating and deploying one such NLI has so far proven successful in attracting attention, and usage from enterprises in unprecedented proportions.

References

- [1] Bill Lange, Senior Analyst, Delphi Group, Boston, MA "One Step Interaction with CRM Applications", Snapshot, August 2002.
- [2] L. Ahrenberg, N. Dahlback, A. Jonsson, A. Thuree, Customizing Interaction for Natural Language Interfaces, Computer and Information Science, Vol. 1, No. 1, 1996.
- [3] B. Hodjat, M. Amamiya, Applying the Adaptive Agent Oriented Software Architecture to the Parsing of Context Sensitive Grammars, IEICE Transaction on Information and System, Vol.E83-D, No.5, pp. 1142-1152, Japan, 2000.
- [4] N. R. Jennings and M Wooldridge (2000) "Agent-Oriented Software Engineering" in Handbook of Agent Technology (ed. J. Bradshaw) AAAI/MIT Press.
- [5] Introducing the Adaptive Agent Oriented Software Architecture and its Application in Natural Language User Interfaces, Hodjat B., Amamiya M., IJCAI Work-shop on Distributed Constraint Reasoning held at IJCAI'2001, pp.109-122, Seattle, 2001.

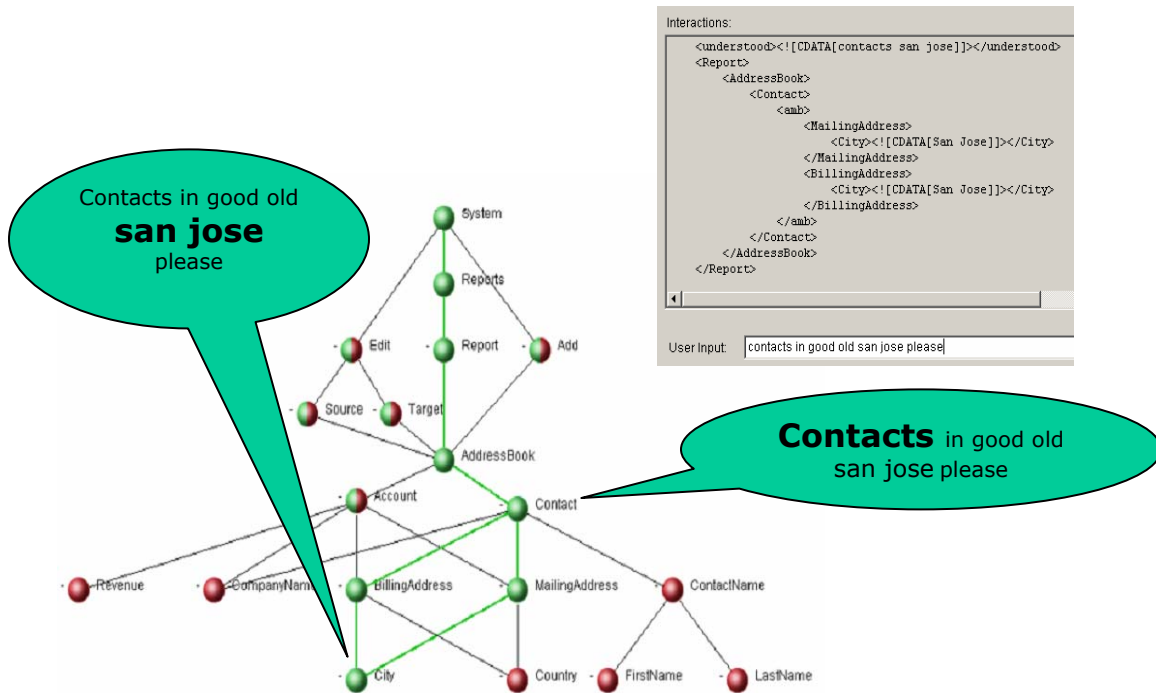


Figure 1) An Agent network is a semantic model of an application. Agents make claims that merge or compete for a role in serving the user. Claims are made based on patterns in the user request, as well as dialog history, application state, user privileges, user preferences, customizations, and other contextual clues.

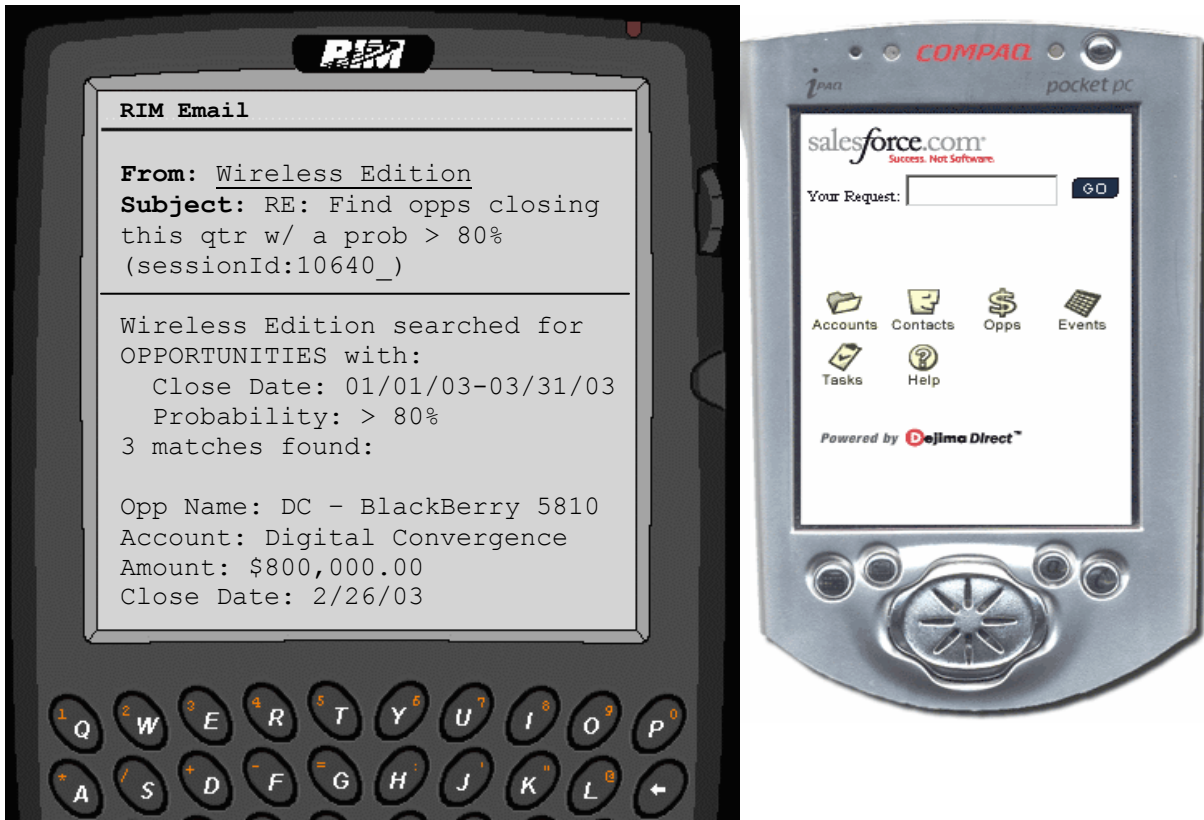


Figure 2) The Wireless Edition supports both an e-mail modality, which is useful for batch or online/offline modes, and a wireless-web modality, which allows a combination of text and clicks.

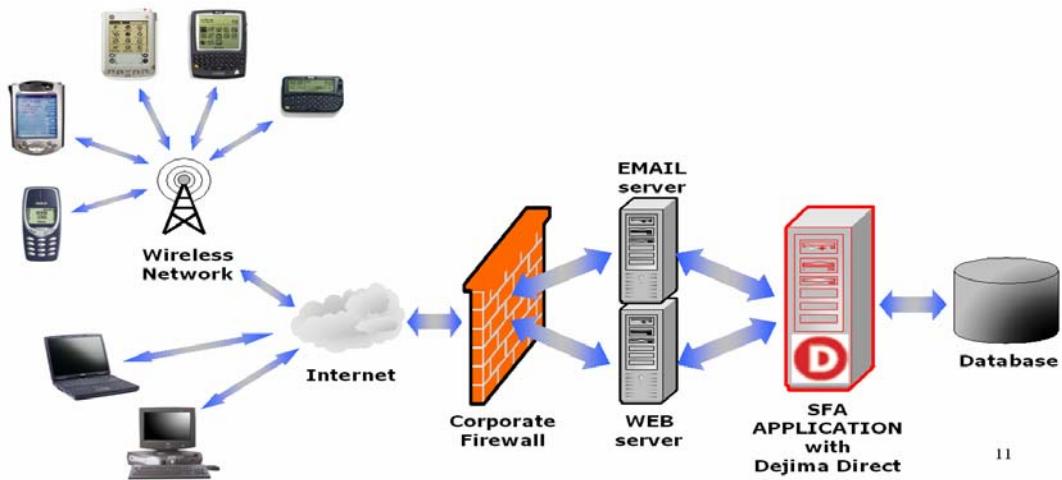


Figure 3) The Wireless Edition is deployed on the server side providing a device agnostic service with no client software requirements.

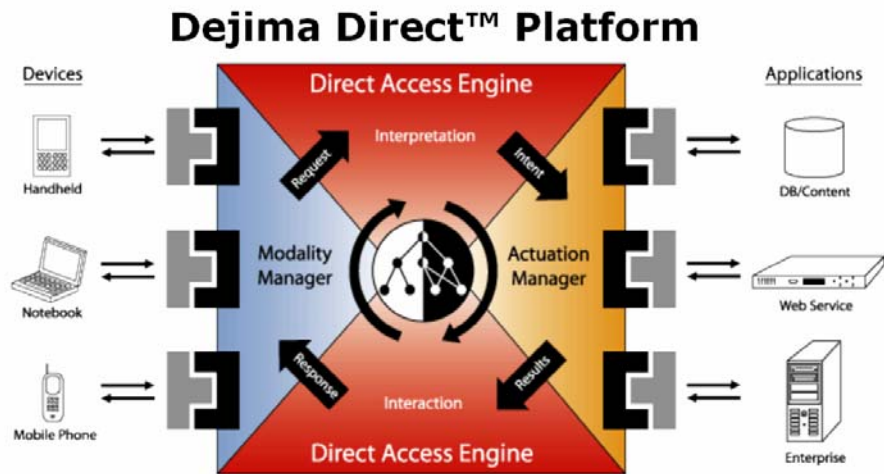


Figure 4) The Dejima Direct Platform consists of an agent-oriented core that connects to front-end devices and backend services through adapters that feed different modality information into the agent network, or convert a generic representation of the user intent to service API calls on the backend.