From Desktop to Mobile: Adapting a Successful Voice Interaction Platform for Use in Mobile Devices

Nikos Tsourakis
ISSCO/TIM/ETI
University of Geneva
Switzerland
Nikolaos.Tsourakis@issco.unige.ch

Agnes Lisowska
ISSCO/TIM/ETI
University of Geneva
Switzerland
Agnes.Lisowska@issco.unige.ch

Pierrette Bouillon
ISSCO/TIM/ETI
University of Geneva
Switzerland
Pierrette.Bouillon@issco.unige.ch

Manny Rayner
ISSCO/TIM/ETI
University of Geneva
Switzerland
Emmanuel.Rayner@issco.unige.ch

ABSTRACT
Even though natural language voice-only input applications may be quite successful in desktop or office environments, voice may not be an appropriate input modality in some mobile situations. This necessitates the addition of a secondary input modality, ideally one with which users can express the same amount of content as they can with natural language using a similar amount of effort. The work presented here describes one possible solution to this problem - leveraging existing help mechanisms in the voice-only application to support an additional non-voice input modality, in our case text input. The user can choose the speech or text modality according to their current situation (e.g. noisy environment) and have the same interaction experience.

Categories and Subject Descriptors
H.5.2 **[Information Interfaces and Presentation]**: User Interfaces – *graphical user interfaces (GUI), natural language, voice I/O*

General Terms
Design, Human Factors

Keywords
Intelligent help system, Mobile multimodal applications, Regulus

1. INTRODUCTION
Voice interaction has been shown to work quite well for domain-specific desktop applications developed using the Regulus platform [1]. These applications are developed such that voice (and more specifically natural language) is the integral input modality. A GUI is used to display recognition and retrieval results and a pointing device such as a mouse is available, but acts as a passive input modality used only to make selections in lists or to confirm actions. It is the voice interaction that provides the contentful – active –interaction.

MobileHCI 2008, September 2–5, 2008, Amsterdam, the Netherlands.

However, voice applications used on mobile devices (whether it be laptop computers, PDAs or mobile phones) face a problem that is of less concern in desktop environments - voice may not always be an *appropriate* input modality.

In certain situations - for example during a business meeting or in noisy environments such as a train - it may not be appropriate to use voice input (meeting) or possible to use it successfully (noisy environment). Consequently, an additional way to provide input with the same amount of content is needed. The challenge is to do this while minimally:

1. Affecting the screen real-estate used.
2. Increasing the number of functionalities that need to be added/learned by the user.
3. Modifying a back-end infrastructure (Regulus) that has already been show to be successful.

The goal of this paper is to describe our approach to resolving this problem in the framework of one of the areas being explored using Regulus on a mobile device - a calendar application in which users can ask questions about dates, locations and participants of meetings that have already occurred or are planned. In the rest of this paper we first briefly explain how applications developed on the Regulus platform work. Then we go on to discuss our proposed solution in both theory and in practice, and we conclude with some future work to evaluate the implementation.

2. REGULUS IN FIXED DEVICES
Regulus is an Open Source platform that supports construction of rule-based medium-vocabulary spoken dialogue applications. It has already been used to build several substantial speech-enabled applications, of which the most prominent are NASA's Clarissa procedure navigator [2], which reached to the point of initial testing on the International Space Station[1]; and Geneva University's MedSLT medical speech translator [3], which was successfully used by medical students.

The distinguishing feature of the system is the emphasis on principled use of linguistically motivated methods; all speech and language processing is performed using resources ultimately derived from substantial, domain-independent feature grammars, suitably compiled for the tasks of analysis, generation and speech recognition. Early versions of the platform used one base

[1] http://ic.arc.nasa.gov/projects/clarissa/iss_report.html

grammar per language; more recently, we have even proceeded beyond this point, and merged together the resource grammars for related languages [4]. Compilation of the general grammar into its final form proceeds in several stages, and involves example-based methods, driven by small corpora, which make it possible to transform the loose general grammar into tightly constrained domain-specific grammars. For the case of recognition, subsequent processing compiles the domain-specific grammar into a Grammar-Based Language Model (GLM) in Nuance format.

In a spoken dialogue application it is often desirable to give users immediate feedback on the system's coverage (the type of language that the system can 'understand'), which is particularly likely to be useful when recognition has been unsuccessful. This can be offered through an intelligent help module, which tries to identify a user's request and propose in-coverage alternatives. Work by Hockey at al. [5] suggests that most users are able to use this kind of feedback to gain rapid familiarity with the system's coverage.

The help module for a system built on the Regulus platform (for voice-only input) works in the following way. The system is provided with a library of examples, compiled during system development, which are known to be within coverage. At runtime, the application carries out a second round of recognition using a backup recognizer equipped with a Statistical Language Model (SLM); and by using the contents of the library displays the closest matches to the user. This happens independently of whether the user's original input was correctly recognized or not, which means that for each input, the user always sees a set of related in-coverage phrases.

From our informal observations the SLM seems to be more robust compared to the GLM for obtaining the help sentences. When performing recognition with the GLM the system tries to recognize complete, well structured sentences. Therefore the output of the GLM maybe far from the user's input especially if this input is out of coverage. Instead we prefer to identify semantically important words in the N-gram contained in the SLM.

We showed in [6, 7] that the help module makes a significant difference in interaction. With help available, most users rapidly become confident in their ability to use the system; without it, they flounder. Table 1 shows some help examples for the input sentence: "Do you know what meetings there are next week?".

Table 1. Help System.

User	Do you know what meetings there are next week?
System	Is there a meeting in the next week? What meetings are there during the next week? What meetings do we have next week? What meetings will there be next week? What meetings are there next week?

In the following section we explain our approach to adding additional contentful input modalities to this environment.

3. ADDING SUPPORT MODALITIES TO REGULUS APPLICATIONS

As it became obvious, for the reasons described in the introduction, that another way of giving contentful input would be needed when Regulus-based applications were deployed on mobile devices, the question that loomed was which type of input would be best.

Regulus-based applications have three useful properties that we found greatly influenced our choice. The first is the ability to quickly process rich natural language input. The second is the way in which the help system guides users as to its coverage. The third, which has only been mentioned in passing thus far, is that because they are primarily dialogue-based systems, the GUI components that are present in Regulus-based applications are quite simple.

For the calendar application (running on Nokia N800), as an alternative means of input to the speech modality we chose text entry via an onscreen keyboard and a stylus, which are built into the device. Choosing the onscreen-keyboard has two advantages - the first is that it is readily available in the device, and the second is that users have the possibility to enter exactly the same semantic input using two very different input modalities. Moreover, this solution not only addresses all of the problems outlined in the introduction, but also maintains and leverages the interesting properties we described in the previous paragraph.

The GUI and use of screen real-estate remains virtually unchanged from the voice-only version since the onscreen keyboard can be called up on an as-needed basis and cover only the part of the GUI that does not need to be directly visible while input is being provided. The keyboard can then disappear once the user has finished entering the input, reverting to the same GUI as would be visible during voice-only interaction. This solution also addresses the problem of increasing functionalities since the user would only need to tap on the input bar (which is also present in the voice-only version for showing recognition results) with the stylus to call up the keyboard.

The issue of minimally modifying the back-end infrastructure when incorporating text input is a bit more complicated. In Regulus applications, it is assumed that the user says well formed (full or elliptical) sentences, whose terminology and structure are covered by the grammar. Consequently, all sentences which are correctly recognized by the GLM can be analysed and processed. If a sentence is not in-coverage and cannot be correctly recognized, the help system intervenes and proposes examples of sentences that are possible and in-coverage for the system, as explained in the previous section. In such cases, the speech input is converted by the SLM into a chain of words which are recognized by the system, which makes it possible to compare them to the library of example sentences. However, when input is given via text, there are two essential differences to consider.

The first is that work with the Archivus system[2] showed that users were most likely to use keywords when entering input with text rather than short expressions or full sentences [personal communication]. Table 2 shows the distribution of the different types of text input observed with the Archivus system. However, the Regulus grammars are not capable of analyzing strings of keywords.

[2] A multimodal desktop application to access multimedia meeting data [9] using voice or text for linguistic input and mouse or pen for pointing.

Table 2. Distribution of types of text input.

Keywords	Short expressions	Full sentences
59%	28%	13%

The second problem is that input can be malformed (typos, spaces, punctuation) from the perspective of the grammar analyzer, resulting in words not being recognized by the application.

The way that we saw to solve this problem was to leverage the efficacy of the existing help module by extending it to handle text input. The goal is to guide the user to the types of text input that the system can process. How we do this is described in the following sections.

4. REGULUS MOBILE SYSTEM ARCHITECTURE

The system [8] uses a distributed architecture, where the various nodes are configured as autonomous peers in the same network, and offer different kinds of services. The mobile device, which is the only part the user sees, contains all the logic needed to communicate with the other peers. Figure 1 shows the top-level components of the network.

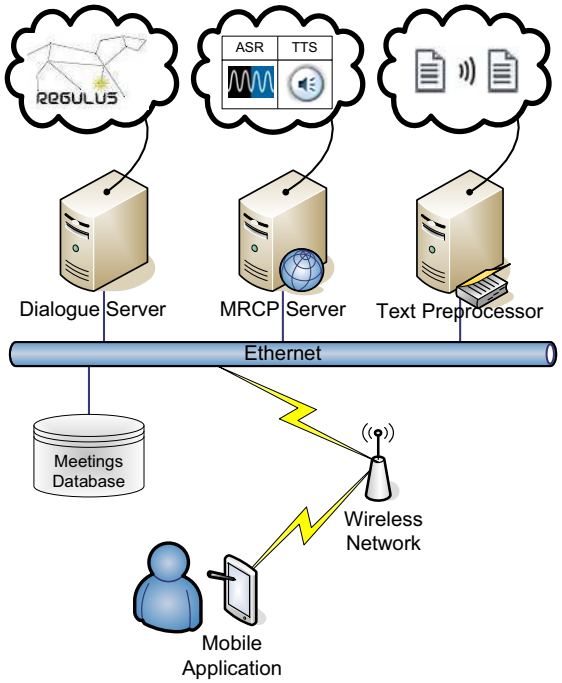

Figure 1. Mobile System Architecture

When the user speaks to the device, audio packets are transmitted through the wireless network to the ASR server, where they are recognized using both the grammar-based and the statistical recognizers. The recognition results, in the form of N-best speech hypothesis lists, are sent to the dialogue server. This performs all necessary natural language processing; its output is the dialogue response (the meeting data), together with a set of help sentences.

We extended this work in order to include a new node called the Text Preprocessor. This node is responsible for handling any text input from the user for further processing and is described in the following sections.

4.1 Mobile application

The mobile application is a lightweight process, implemented in C++, responsible for the following tasks:

- Supporting different input modalities (speech, pen buttons) and different output modalities (screen and speakers).
- Communicating and requesting services from the Media Resource Control Protocol (MRCP [10]) server for either ASR or TTS.
- Capturing and packetizing the audio (8KHz - 8bit) using the Real-Time Transport Protocol (RTP).
- Forwarding the recognition result to the Dialogue server in order to perform the natural language processing.
- Providing the answer to a user's request and presenting a set of help sentences according to the user's input.
- Communicating with the Text Preprocessor for transforming the text input into a "valid" query string.

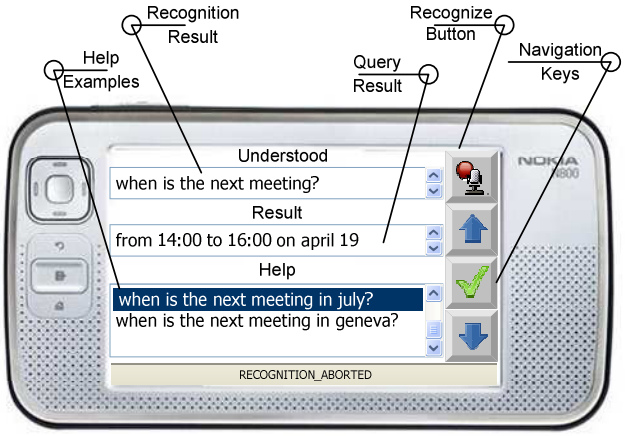

Figure 2. Mobile Calendar GUI

As seen in Figure 2, the Calendar GUI consists of three text boxes for presenting:

- The recognition result (grammar-based recognition) or the text input area.
- The result of the data query.
- A list of help examples associated with the recognition result (statistical recognition).

There is also a button for initiating the speech recognition and three buttons for navigating in the help list. The same functionalities are also offered by the hardware buttons located in the upper left side of the device. Finally, the status bar presents different events associated with the user's interaction (e.g. start of speech, recognition completed etc.).

In the case of text input the user can use the stylus to pick letters from the onscreen keyboard (Figure 3) and submit his/her request to the Text Preprocessor by pressing enter.

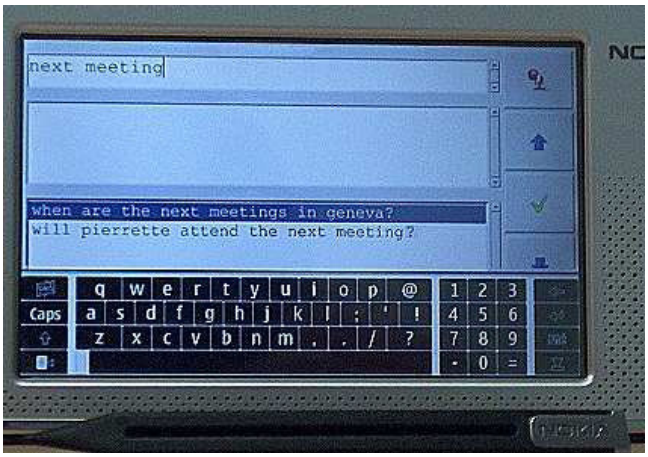

Figure 3. Text Input

4.2 The Text Preprocessor

As has already been mentioned, the nature of the Regulus infrastructure is such that when using voice input the user will always get some feedback, either as a result for their query, or help meant to guide them towards in-coverage phrases.

However, the case for the text input is different. If the text cannot be processed successfully in the context of the SLM, no result and no help is produced since the output of the dialogue server relies on the results of this processing. Therefore, a mechanism for transferring the original text into the context of the SLM is needed. In order to accomplish this the original text is converted into speech and then back to text, hence the name text pre-processor.

The idea is straightforward; when the user types a text string use the text preprocessor module to:

1. Perform a cleaning phase e.g. remove any special characters (commas etc.). Send it to the dialogue server and wait for the response (query result and help sentences). If the input cannot be parsed proceed to the next step.
2. Send the input text to the text-to-speech server. Obtain the audio packets and perform recognition using the SLM model.
3. After completing step 2 the initial text string has been converted into a valid form. Send it to the dialogue server to obtain the set of help sentences presented to the user.
4. The user can pick one of the help sentences in order to perform a data query or to rephrase it according to his needs.

While it might be feasible to create modules to do natural language processing on the text to replace the need to covert text into speech and then back again, another benefit of this approach is that constraints on orthography do not need to be very strict. A minor orthographic mistake will be eliminated most of the time when the text is pronounced by a TTS. By performing recognition on the resulting audio the user will be more likely to get a result close to what they were looking for than if they were using a more standard text processing module. Even if the output is far from the user's request they will be introduced to the application's coverage and use it during the next interactions.

5. FUTURE WORK

The next step of the project will be to evaluate whether the introduction of text input as a secondary semantic input modality in this way is both helpful and acceptable to users. In order to do this, we will run an experiment, looking at two results in particular:

1. whether users are equally successful using text and voice input
2. whether and to what degree the help system influences interaction over time

5.1 Description of the experiment

The experiment will involve 10 users with approximately the same level of experience with mobile devices and voice or stylus input. Each user will use both versions of the application (voice-only input and text-only input) two weeks apart. The time lag is intended to minimize the effect of learning the system coverage. The approximate time taken to do the experiment with the first version of the application should not be long enough for the user to be able to remember the system coverage when they are asked to do the experiment with the second version of the application. Half of the users will start with the voice-only system and the other half with the text-only system.

Each user will be asked to find the answers to a series of questions, where the questions will be presented in situational paragraphs such as this example:

"Nikos is sitting on a train on the way to a meeting in Geneva, but he doesn't remember which room the meeting is in, or who will be attending. Find this information."

Such paragraphs give the users enough contextual information to pinpoint the relevant meeting in the database as well as specifying the information that they need to find, while minimally influencing the linguistic structure and vocabulary they use.

5.2 What is being investigated

When looking at whether users are equally successful using text and voice input, success will be measured objectively in terms of task completion time and the number of interactions, or turns, needed to find the answer to a particular query.

In order to examine to what degree the help system influences interaction over time, we will look at

- whether text input becomes increasingly longer and linguistically structured further into the experiment and,
- for both text and voice input, whether the vocabulary and linguistic constructions that users are using increasingly follow those used in the help system.

We believe that the presence of both of these factors in the data implies that the help system is influencing the type of input that the users are producing.

6. CONCLUSIONS

In this paper we presented a solution for adding a second equally expressive input modality to a Regulus-based voice-only natural language driven application ported on a mobile device. We were able to add text input via stylus and on-screen keyboard with minimal modification to both the existing voice-only interface and the backend infrastructure of the application. This was done by leveraging the advantages of the powerful help system available in the voice-only applications. The help system was modified so that it could give the same type of help independent of whether linguistic input was provided in voice or text form, which we believe will help users learn the linguistic coverage of the system faster. This in turn would shorten the amount of time needed to achieve smooth interaction with the system.

7. REFERENCES

[1] Rayner, M., Hockey, B.A. and Bouillon, P. 2006. Putting Linguistics into Speech Recognition: The Regulus Grammar Compiler. Stanford, California, CSLI Press.

[2] Rayner, M., B. A. Hockey, N. Chatzichrisafis, K. Farrell and J.-M. Renders. 2005. A voice enabled procedure browser for the International Space Station. Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL), Ann Arbor, Michigan.

[3] Bouillon, P., M. Rayner, N. Chatzichrisafis, B. A. Hockey, M. Santaholma, M. Starlander, H. Isahara, K. Kanzaki and Y. Nakao. 2005. A Generic Multi-Lingual Open Source Platform for Limited-Domain Medical Speech Translation. Proceedings of the 10th Conference of the European Association of Machine Translation, Budapest, Hungary.

[4] Bouillon P., Rayner M., Novellas Vall, B., Starlander M., Santaholma M., Nakao Y. and N. Chatzichrisafis. 2007. Une grammaire partagée multi-tâche pour le traitement de la parole : application aux langues romanes, Traitement Automatique des Langues, Volume 47, 3/2006, Hermes & Lavoisier.

[5] Hockey B.A., Lemon O., Campana E., Hiatt L., Aist G., Hieronymus J., Gruenstein A., and Dowding J. 2003. Targeted help for spoken dialogue systems: Intelligent feedback improves naive user's performance. In Proceedings of the 10th EACL, Budapest, Hungary.

[6] Chatzichrisafis N., Bouillon P., Rayner M., Santaholma M., Starlander M., and Hockey B.A., 2006. Evaluating task performance for a unidirectional controlled language medical speech translation system. In Proceedings of the HLT-NAACL International Workshop on Medical Speech Translation, pages 9–16, New York.

[7] Rayner M., Bouillon P., Chatzichrisafis N., Hockey B.A., Santaholma M., Starlander M., Isahara H., Kanzaki K., and Nakao Y. 2005. A methodology for comparing grammar-based and robust approaches to speech understanding. In Proceedings of the 9th International Conference on Spoken Language Processing (ICSLP), Lisboa, Portugal.

[8] Tsourakis, N., Georgescul M., Bouillon P. and Rayner M. 2008. Building Mobile Spoken Dialogue Applications Using Regulus. To appear in the LREC 2008, Marrakech, Morocco.

[9] Ailomaa, M., Lisowska, A., Melichar, M., Armstrong, S., and Rajman, M. 2006. Archivus: A multimodal system for multimedia meeting browsing and retrieval. Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions. Sydney, Australia July 17th-21st, 2006.

[10] Shanmugham, S., P. Monaco and B. Eberman (2005). A Media Resource Control Protocol (MRCP) Developed by Cisco, Nuance, and Speechworks. Internet Engineering Task Force.