A Mobile Fitness Companion

Olov Ståhl[1], Björn Gambäck[1, 2], Preben Hansen[1], Markku Turunen[3] and Jaakko Hakulinen[3]

Abstract. The paper introduces a Mobile Companion prototype, which helps users to plan and keep track of their exercise activities via an interface based mainly on speech input and output. The Mobile Companion runs on a PDA and is based on a stand-alone, speaker-independent solution, making it fairly unique among mobile spoken dialogue systems, where the common solution is to run the ASR on a separate server or to restrict the speech input to some specific set of users. The prototype uses a GPS receiver to collect position, distance and speed data while the user is exercising, and allows the data to be compared to previous exercises. It communicates over the mobile network with a stationary system, placed in the user's home. This allows plans for exercise activities to be downloaded from the stationary to the mobile system, and exercise result data to be uploaded once an exercise has been completed.

1 INTRODUCTION

A computational agent that acts as a conversational partner to its user, builds a long-term relationship to the user, and learns about the user's needs and preferences is termed a 'Companion' [1] or a 'relational agent' [2]. If such a Companion is to be persistent and totally integrated into the life of the user, it must be fully mobile and have the ability to take different shapes and forms in different situations. In essence, building a truly mobile Companion requires us to address three basic issues in the design of conversational agents:

- What physical form(s) can a mobile Companion take?
- How can the computer system itself be adapted to the (limited) size of the physical mobile platform?
- How does the mobile setting influence the user interaction?

In the present paper we concentrate on the last two issues, while touching the first one. However, for the sake of the present discussion we will here limit ourselves to physical forms for a mobile Companion that are already available on the market, that is, mobile devices such as cell phones, PDAs, etc.

1.1 The COMPANIONS Project

The overall ambitious goal of the EC-funded (FP6/IST) project COMPANIONS (www.companions-project.org) is to develop autonomous, persistent, affective and personal multimodal interfaces with robust dialogue capabilities (based on machine learning strategies). To this end, fifteen partners from eight different countries in Europe and the US will collaborate over a 4-year period. At this time, the project is barely half-way through, but two such conversational agent prototype systems have been built [3]. One aims to enable dialogues about the user's photographs, while the other wants to support the user in upholding a healthy lifestyle.

1.2 The Health and Fitness Companion

The second current Companion prototype, the Health and Fitness Companion, is shown in Figure 1. The Companion has a stationary ("home") component which accounts for the main part of the user interaction, and a mobile component which is the topic of the present paper. The system components communicate with each other over a regular mobile phone network. The overall prototype is described in detail in [4].

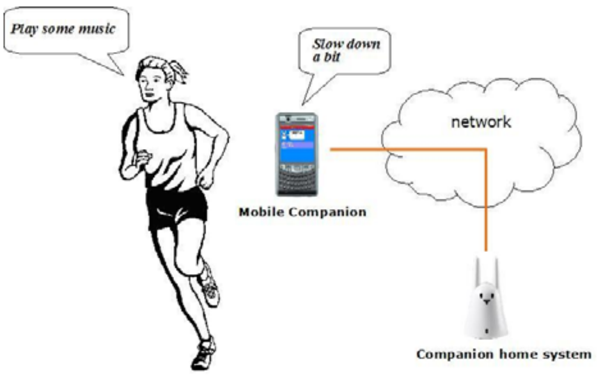


Figure 1. Overview of the Health and Fitness Companion

It is important to note that the overall aim of the Health and Fitness Companion prototype is to develop the Companion concept, rather than to study the actual fitness area as such. Thus it is not of vital importance that the system should be a first-rate fitness coach, but it is essential that the system should be able to take a persistent part in the user's life, that is, that it should be able to follow the user in all the user's activities. This means that the Companion must have mobile capabilities. Not necessarily self-mobile (as a robot or a pet), but mobile so that the user can bring the system with her, like a handbag or a pair of shoes. Or as a mobile phone or PDA, which is what the present Mobile Health and Fitness Companion is all about.

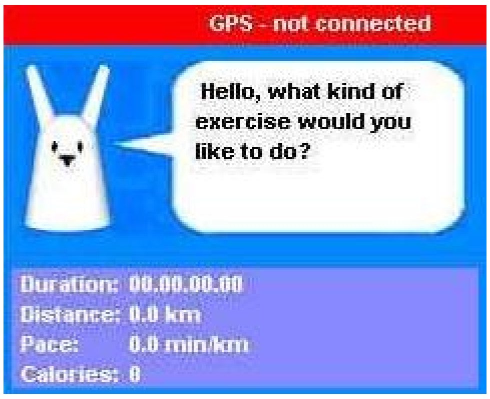
The rest of the paper is laid out as follows: First the Mobile Companion prototype as such is described. Then we will go into detail on the user interaction with the Mobile Companion (in Section 3) and the actual implementation (Section 4). Section 5 airs some initial user evaluation topics, while Section 6 discusses related work by others. Finally, Section 7 sums up the paper and points to some directions in which the work can be extended.

[1] SICS, Swedish Institute of Computer Science AB.
`{olovs,gamback,preben}@sics.se`
[2] Department of Computer and Information Science, Norwegian University of Science and Technology (NTNU), Trondheim.
[3] Department of Computer Science, University of Tampere.
`{mturunen,jh}@cs.uta.fi`

2 THE MOBILE COMPANION PROTOTYPE

The Mobile Health and Fitness Companion prototype runs on a mobile handset (e.g., a PDA), and can be used during physical exercise (e.g., while running or walking) to track the distance, pace, duration, and calories burned. The data gathered during an exercise is stored in the device's record store, and can be used to compare the results to previous or future runs. The interaction with the user is supported by a user interface that allows input to be provided via speech, buttons or stylus interaction, while system output to the user can be purveyed via speech as well as in text messages on the graphical display.

The Mobile Companion communicates with the stationary Companion (home-system) that presently takes the physical form of a Nabaztag rabbit (www.nabaztag.com). To give the user a feeling of Companion persistence, the user interface of the Mobile Companion shows an image of a Nabaztag rabbit along with some text areas where various exercise and device status information is displayed. The rabbit has a speech bubble, which is used to present textual versions of all sentences spoken by the Companion via TTS. The interface is made up of a single screen, as shown in Figure 2.


GPS - not connected

Hello, what kind of exercise would you like to do?

Duration: 00.00.00.00
Distance: 0.0 km
Pace: 0.0 min/km
Calories: 0

Figure 2. The Mobile Companion GUI

At the top of the screen is a red status bar that shows the status of the GPS device that is used to track the user's movements during the exercise. The application is able to use an external Bluetooth GPS, or a GPS receiver that is built into the PDA device.

Below the GPS status bar is the avatar and text output area, showing the Nabaztag rabbit image and a speech bubble. The rabbit image is static, that is, no animation is performed, for instance as TTS output is produced. The speech bubble is used to output the same text as is spoken by the Companion via TTS. By using the rabbit as the visual representation of the Companion, we try to make a connection to the physical Nabaztag rabbit used by the home system. This will hopefully give the users a sense of communicating with the same Companion, no matter if they are using the stationary or mobile system. To further the feeling of persistence, the stationary and mobile parts of the Health and Fitness Companion also use the same TTS voice.

The exercise status bar at the bottom of the screen shows the current status of the exercise at a glance. The exercise information that is available is: duration (how long the exercise has been going on), the distance covered so far, the current pace (an average over the last ten seconds), and calories burned. The Companion can also report the same information (duration, distance, pace, and calories) via speech output. This is done when the user gives a specific voice command or presses a specific button on the device.

3 INTERACTING WITH THE MOBILE COMPANION

The interaction with the Mobile Health and Fitness Companion is based almost entirely on the use of speech. This means that the Companion will use synthesized speech to address the user (typically ask questions), and the user will reply by speaking into the microphone. However, the user can interact with the Mobile Companion in three different ways:

1. **Via voice**: The normal mode of operation. The mobile prototype is capable of doing TTS and speaker-independent ASR, with a push-to-talk model.

2. **Via button presses**: Some of the PDA device's keys can be used as shortcuts to input commands, for instance, to request the Companion to summarize the status of an ongoing exercise. The same command can also be given via voice.

3. **Via stylus interaction**: If the PDA is equipped with a touch sensitive screen, input to some questions asked by the Companion can be answered by tapping on a list of possible choices on the screen. The same input can also be given via voice.

When the application is started, the user is presented with a greeting. The Companion then asks whether it should connect to the home system and download the current plan. Such a plan consists of various tasks (e.g., shopping or exercise tasks) that the user should try to achieve during the day, and is generated by the home system during a session with the user. If the user chooses to download the plan – and the download is successful – the Companion summarizes the content of the plan for the user, excluding all tasks that do not involve some kind of exercise activity. The Companion then suggests a suitable task based on time of day and the user's current location. If the user chooses not to download the plan, or rejects the suggested exercise(s), the Companion instead asks the user to suggest an exercise. Once the exercise type is agreed, the Companion checks the status of the GPS receiver, and outputs a warning message if the user's current position cannot be determined. This happens, for instance, if the user is still in-doors. The Companion is then waiting for the user to initiate the exercise, which can be done via voice or by pressing a specific button on the mobile device.

During the exercise, the user is able to give a few commands at any time:

- Ask for exercise status: The Companion will reply with a summary of the exercise. If the GPS is not working, the data will be restricted to the duration.
- Ask the Companion to start playing music, to move on to the next song, or to stop playing music.
- Stop the exercise. The Companion will confirm this and then give a summary of the exercise results.

After the exercise is over and the result summary has been given to the user, the Companion asks whether the result should be uploaded (to the home system). If the user agrees, the Companion sends the information to a web server, where it can be downloaded and examined by the home system later on.

The error recovery in the present version of the Mobile Companion is fairly rudimentary. To avoid misunderstandings, the Companion requests confirmations concerning user input if the ASR confidence score is below a certain threshold. Thus:

C: What kind of exercise would you like to do?
U: I would like to go for a walk
C: You have selected walking, correct?

If the user says something not defined by the current grammar, the Companion tells the user that it cannot understand the input. If, for instance, a silence timeout occurs (the user pushes the "talk" button but does not say anything), the Companion tells the user that it could not hear what the user said. Should the Companion be expecting an answer to a question, the question is then repeated after a short pause.

4 IMPLEMENTATION

The Mobile Companion currently runs on a Windows Mobile-based Fujitsu Siemens Pocket LOOX T830 device, which has a 416 Mhz XScale processor (128 MB RAM) and a built-in GPS receiver. The prototype is made up of two programs:

1. A Java midlet (MIDP2) that controls the main application logic (exercise tracking, dialogue management, etc.), as well as the graphical user interface.
2. A C++-based speech server that performs TTS and ASR functions on request by the Java midlet.

The midlet and the speech server communicate via a TCP socket. The socket is used to send "commands" from the midlet to the speech server, and to send back results in the other direction. The following sections describe the two programs in more detail.

4.1 Java midlet

The main application is based on Java technologies such as PART (Pervasive Applications RunTime; part.sourceforge.net). The midlet is made up of a number of singleton manager Java classes that provide various basic services (event dispatching, GPS input, audio playback, TTS and ASR, etc.). However, the main application logic is implemented using the Hecl scripting language (www.hecl.org).

When the midlet is started, it reads a script file from the mobile device's file system. The script file is evaluated in a Hecl interpreter. The script then builds the graphical user interface by using various commands for rendering images, drawing text, and so on. The script also has access to a number of script commands defined by the various Java manager classes, allowing it to initiate TTS and ASR operations, etc. Events produced by the Java code are dispatched to the script, which means that it will be notified about, for instance, the user's current GPS position, GUI interactions (e.g., stylus interaction and button presses), and voice input (Figure 3).

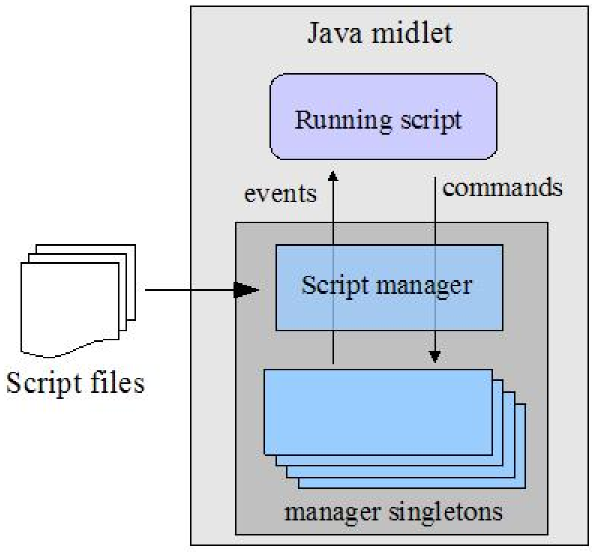


Figure 3. The interaction between the script and Java managers

One of the services supplied by the Java manager classes is to provide a persistent data store where information can be saved in-between Companion sessions. This allows information that is produced during a session to be accessed and examined in a later session. The information is represented by PART objects to which dynamic properties can be added. Such properties consist of `<name,value>` tuples, where both `name` and `value` are strings:

```
obj.addProperty(name,value);
String val = obj.getProperty(name);
```

These objects can be saved and loaded using the midlet's data store manager, which in turn uses one of the persistency managers provided by PART. Currently the store is mapped to the device's record store (a concept provided by J2ME).

The object store is also available to the application script via a number of scripting commands added by the data store manager. Since the scripting language does not support object-oriented programming, an object reference parameter is used instead to identify which object the command concerns:

```
set objRef [objcreate]
setprop objRef name value
set value [getprop objref name]
```

In the current implementation, the data store is used to save the result of exercises and various information about the user, such as name, weight, and preferences.

4.1.1 Dialogue management

The Companion uses a push-to-talk model for speech input. When the user wants to say something to the Companion, a button needs to be pressed to indicate that speech input will follow. The button press causes the Java code to send a command to the speech server to activate speech recognition.

The dialogue with the user is implemented in the scripting language using a state machine model. Each state is represented by a scripting procedure that receives and analyzes the user's input while the dialogue is in that particular state. Based on the analysis, the active input procedure decides whether to move on to another state, or to stay in the current one. A state transition is achieved by setting the target state's input procedure as the current one, making it responsible for handling user input the next time it occurs. State transitions are often combined with some speech output, a change of the voice recognition grammar, and in some cases, updates to the data store. The script can perform all of these actions by calling various script commands. All input procedures take three arguments:

1. A string representation of the user's input, i.e., the word(s) spoken by the user (according to the ASR).
2. The recognition confidence, i.e., the likelihood that the recognized input string matches the actual users input.
3. The semantic expression returned by the grammar rule matching the user's input (may be null).

An example script input procedure is shown below. It handles user answers to the system questions such as "You have selected jogging, correct?"

```
proc confirmExSel {input confidence semExpr} {
  if {eq $semExpr "CONFIRM"} {
    asr_grammar_select "exerciseCommands"
    set_asr_input_handler await_exercise_commands
  } elseif {eq $semExpr "REJECT"} {
    tts_question "What kind of exercise would
                  you like to do then?"
    asr_grammar_select "exerciseSelect"
    set_asr_input_handler awaitSelectExercise
  }
}
```

If the user gives a positive answer (e.g., "Yes" or "Correct"), the grammar will return the semantic tag CONFIRM, which will end up as the value of the *semExpr* parameter. In the same way, if the user gives a negative answer (e.g., "No" or "That is not correct"), the value of the *semExpr* parameter will be REJECT. If the user confirms the exercise selection, the script changes grammar and moves on to a state where exercise commands are handled. Otherwise, a TTS message is generated, the grammar is changed and the script moves on to a state expecting the user to select another type of exercise.

Note that only the most likely hypothesis is sent from the speech server to the midlet, and then delivered to the active script input procedure, that is, there is currently no way for the input procedure to access the "next" hypothesis (if available) – or an n-best list from the ASR – should it not be able to make sense of the current input hypothesis.

In the current implementation, all semantic expressions used by the grammars are in fact string literals, simply because an earlier version of the ASR library that was used supported no other semantic information. However, the library has now been upgraded to a version supporting full ECMA scripts, which will be utilized in future versions of the Mobile Companion.

Currently, all system reactions to the user input are based on the semantic information only (i.e., the input string is ignored). As a result, it is possible for the user to also give input via button presses as well as stylus interaction. When the user produces voice input, the input information (input string, confidence and semantic expressions) is delivered to the application script from the Java code via an *ASR input event*. Such events are delivered to a specific script event handler, which extracts the input information and then calls the active input procedure. These events can also be produced by the script code itself, so the script programmer has the option to map certain button presses and screen interactions into input events, which will then be treated as "ordinary" input events by the rest of the system. As an example, during an exercise the user has the choice of giving input either via voice or button presses to control the playback of music or to request status reports.

ASR errors reported by the speech server are passed from the Java code to the application script via an *ASR error event*. Such events will cause the Companion to respond with the spoken message "Sorry, I don't understand". Also, if the Companion is expecting an answer to a question, the question is repeated. ASR errors may, for instance, occur if the user's utterance cannot be recognized using the current grammar, or if the ASR library cannot detect any input long after the recognition was activated.

The dialogue in the current prototype makes use of eleven dialogue states (script input procedures), and seven different input grammars. The total grammar word count is around 100.

4.1.2 Integration with the home system

The plan generated by the home system exists as an XML document available on the web. When the Mobile Companion is started, it asks the user if the plan should be downloaded (if not already done in a previous session the same day). If the user agrees, the plan XML document is downloaded via an HTTP call to the web server where the plan is stored. This requires the mobile device to have Internet access, for instance, via WLAN or 3G/GPRS. If the download succeeds, the XML document is parsed and all information concerning planned exercise activities is extracted. Such information consists of the type of the exercise (e.g., walking or cycling), possible start and end points (e.g., from home to work), time of day (e.g., during lunch), and so on. Based on the time of day and the user's current location, the Companion suggests an exercise to the user (given that a "matching" exercise can be found). If the user declines the Companion's suggestion, it tries to find another exercise to suggest, and so on. If the plan does not contain any exercise tasks – or if the user rejects all suggestions – the Companion will move on and instead ask the user to suggest an exercise.

When the user has completed an exercise that was part of the daily plan, the Companion asks the user if the result (distance travelled, duration and calories burned) should be uploaded to the home system. If the user accepts, the Companion makes a new HTTP call to the same web server, supplying the exercise result as part of the URL. The result will be saved by the web server as another XML document, which can be accessed by the home system and referred to in later sessions.

4.2 Speech server

When the speech server is started, it creates a socket endpoint and then waits for the Java midlet to connect. Once a connection has been established, the server accepts a number of "commands" which can be sent by the client over the socket. The server also sends back messages to the client, for instance, notifications about certain speech-related events, or command results. The following set of commands is available:

- **VOICE_SELECT** *nameOfVoice* – set the voice that will be used for speech output.
- **TTS** *message* – speak a message using the Text-to-Speech synthesizer.
- **GRAMMAR_LOAD** *nameOfGrammar* – load a grammar that will be used for speech recognition.
- **ASR** – activate speech recognition.

The events or results that are sent back from the speech server to the Java midlet are the following:

- **TTS_STARTED** – speech output is started as a result of the server receiving a **TTS** command.
- **TTS_STOPPED** – the speech output has ended.
- **ASR_RESULT** *input confidence semanticTag* – deliver the result of an ASR operation. The *input* parameter is a string representation of the utterance spoken by the user. The *confidence* parameter specifies the speech server's confidence in the returned result. The *semanticTag* parameter is any semantic tag information returned by the grammar rule(s) used to match the user's input.

The speech server may also return a number of error messages, for instance, to inform the midlet that a TTS voice or an ASR grammar was not available, or that a speech recognition operation failed.

The speech server is based on the Loquendo ASR (speaker-independent) and TTS software to handle both speech input and output; currently Loquendo's Embedded ASR 7.4 and Embedded TTS 7.4 systems [5]. The Mobile Companion uses SRGS 1.0 grammars in XML format. The grammars are pre-compiled before being installed on the mobile device.

5 USER EVALUATION

The overall evaluation for the COMPANIONS project is only in its initial stages. As further described in [6], the evaluation will be performed in three phases, with the first one just providing informative feedback on the initial prototypes, while the third phase will be a large-scale analysis of the prototypes, involving more users and increasingly robust prototypes, as well as a more stable experimental evaluation set-up design.

The second phase will play a key-part in that it will expose users to the prototypes for the first time, and enable the evaluators and developers to agree on the key evaluation metrics. Since the Mobile Companion is in an early prototype phase, it is necessary to develop a list of metrics and then collect data. These initial metrics will enable us to achieve three goals:

1. *A first baseline evaluation of system performance* – this is required as a measure of development progress.
2. *Characterisation of the Companion* – this is a mechanism to determine capabilities of the prototype.
3. *Constructive developer feedback*. Developers will be involved in the evaluation and get indications of errors, problems, and issues concerning user feedback.

A set of metrics has been developed for the different evaluation stages: *Speech Metrics* (e.g., word error rate, concept error rate); *Dialogue Metrics* (e.g., dialogue duration, number of turns, word per turn dialogue structure); *Task Metrics* (e.g., task completion); and *User Metrics* (e.g., user satisfaction, requirement elicitation).

Preliminary results from the first phase Mobile Companion evaluation show three key areas that need to be addressed:

- The technology: memory space restrictions; issues regarding using a headset vs. using the PDA as such.
- The user interaction: problems relating to having a speech-driven interface and at the same time interacting with a mobile screen.
- The dialogue: music and dialogue processing interfering with each other.

The third point from the first tests shows that using only the PDA's built-in microphone is not feasible, since the music throws the ASR off track: When the music is playing the system has difficulty in discerning between instructions relating to the exercise and the music control. However, the evaluators indicate that adding the functionality of controlling music choice by voice goes a long way in improving user experience and is very well-suited for a workout scenario.

6 RELATED WORK

As pointed out in the introduction, it is not the aim of the Mobile Health and Fitness Companion to be a full-fledged fitness coach. However, there are several examples of commercial systems that aim to help users with their fitness training via the use of mobile devices, such as miCoach (www.micoach.com) from Adidas and NIKE+ (www.nike.com/nikeplus, www.apple.com/ipod/nike) from Nike. Both systems use a pedometer attached to the user's shoe that communicates speed and distance information wirelessly to a mobile phone or iPod. The miCoach system also uses a heart rate monitor attached to the user's chest. Both systems allow the collected data to be uploaded to a website for further analysis. However, the user interfaces are rather limited in terms of user interaction, and do not aim for a social and emotional relationship with the user, which can be an efficient basis for improving the motivation [2].

MOPET [7] is a PDA-based personal trainer system that supports outdoor fitness activities. The system can guide the user through an exercise session (e.g., jogging), and provides advice based on the current context, a user model and knowledge elicited from a personal trainer and a sport physiologist. The interface is based on a 3D-animated agent that can output text messages as well as pre-recorded audio. MOPET is similar to a Companion in that it tries to build a relationship with the user, but it does not currently use ASR or TTS. There is no real dialogue between the user and the system. Instead the interaction is based on the user pressing buttons on the PDA keyboard to switch between various UI screens, where the agent provides information or requests the user to perform various types of exercise actions.

MPTrain/TripleBeat [8,9] runs on a mobile phone and aims to help users to more easily achieve their exercise goals. This is done by selecting music indicating the desired pace and different ways to enhance user motivation. The system can track the heart-rate and movement (steps per minute) via a user-worn chest-band. This information is then used to select music with specific features that will for instance encourage the user to speed up or slow down. The user interface consists of a number of window screens, for instance displaying exercise information (e.g., pace, heart-rate and calories burned) via text and graphs. The system supports no speech input and output, and does not use an embodied agent user interface model.

InCA [10] is a spoken language-based distributed personal assistant conversational character that runs on a PDA, and that uses a GUI based on a 3D avatar and facial animation. Similar to the Mobile Companion, the architecture is made up of two programs (a GUI client and a speech server), but unlike the Mobile Companion, the InCA server runs as a back-end system on a Linux workstation. This means that audio captures of the user's voice have to be streamed over the network, which introduces delays and requires a constant network connection. A similar solution is presented in [11], although utilizing client-based audio pre-processing using the ES 202 212 standard, allowing speech input to be sent over GPRS connections to the server back-end.

It is thus important to stress that the Mobile Companion is based on a stand-alone, speaker-independent solution, making it fairly unique among mobile spoken dialogue systems, where the common solution is to run the ASR on a separate server, for example [12], and/or to restrict the speech input to some specific set of users/speakers [10].

7 CONCLUSIONS AND FUTURE WORK

We have presented a Mobile Health and Fitness Companion prototype, which helps users to keep track of their exercise activities via an interface based mainly on speech input and output. The Mobile Health and Fitness Companion is currently being developed further in the COMPANIONS project, and we expect the results from the series of user studies currently being carried out to provide several suggestions for modifications and extensions of the prototype.

Apart from input from the user studies, plans for future work include extending the mobile platform with various sensors that can be accessed by the Companion's software. One example is a pulse sensor that gives the Companion information about the user's pulse while exercising, which can be used to provide feedback such as telling the user to speed up or slow down. We are also interested in using sensors to allow users to provide gesture-like input, in addition to the voice and button/screen click input available today.

An extension suggesting itself would be to use the Mobile Companion platform to build a mobile Companion for another domain. As noted in the introduction, the COMPANIONS project is – in addition to the Health and Fitness Companion – also building a Companion allowing for human-computer dialogues centred round digital photographs. A mobile extension of this Companion would, for instance, allow the user to take photographs using the Mobile Companion, have a brief discussion concerning the context of the photograph (current location, who or what is in the picture, etc), and then upload the photograph to the stationary system at home. This would then in addition allow for mobile photo sharing between users, in a fashion similar to the one in [13].

ACKNOWLEDGEMENTS

This work was funded by the European Commission's Information Society Technologies 6th Framework programme under contract number IST-FP6-034434, "COMPANIONS'".

Thanks to Enrico Zovato and Patrizia Pautasso at Loquendo S.p.A. for help with licensing and installation issues regarding the embedded ASR and TTS software. Thanks also to Jay Bradley and Oli Mival from Napier University for initial evaluation feedback.

REFERENCES

[1] Y. Wilks, "Is there progress on talking sensibly to machines"' *Science*, vol. 318, no. 9, pp. 927-928 (2007).

[2] T.W. Bickmore, R.W Picard. Establishing and maintaining long-term human-computer relationships *ACM Transactions on Computer-Human Interaction* vol. 12, no. 2, pp. 293-327 (2005).

[3] M. Turunen, J. Hakulinen, A. Kainulainen, R. Catizone, H. Pinto, G. Gorrell, Y. Wilks, O. Ståhl, B. Tabutiaux, M.C. Rodríguez Gancedo, M. Cavazza, M. Danieli, and D. Pelé, "An initial prototype COMPANION," University of Tampere, Tampere, Finland, COMPANIONS Deliverable D1.1.3, Apr. 2008.

[4] M. Turunen, J. Hakulinen, O. Ståhl, B. Gambäck, P. Hansen, M.C. Rodríguez Gancedo, R. Santos de la Cámara, C. Smith, D. Charlton, and M. Cavazza, "Multimodal Agent Interfaces and System Architectures for Health and Fitness Companions", *4th International Workshop on Human-Computer Conversation*, Bellagio, Italy, 2008.

[5] Loquendo Embedded Technologies: Text to Speech and Automatic Speech Recognition, www.loquendo.com/en/brochure/Embedded.pdf (accessed 2008-09-04).

[6] D. Benyon, P. Hansen, and N. Webb, "Evaluating Human-Computer Conversation in Companions", *4th International Workshop on Human-Computer Conversation*, Bellagio, Italy, 2008.

[7] F. Buttussi, and L. Chittaro, "MOPET: a context-aware and user-adaptive wearable system for fitness training", *Artificial Intelligence in Medicine*, vol. 42, nr. 2, pp. 153-163 (2008).

[8] N. Oliver, and F. Flores-Mangas, "MPTrain: A Mobile, Music and Physiology-Based Personal Trainer", *8th International Conference on Human-Computer Interaction with Mobile Devices and Services*, Espoo, Finland, 2006. ACM.

[9] R. de Oliveira, and N. Oliver, "TripleBeat: Enhancing Exercise Performance with Persuasion", *10th International Conference on Human-Computer Interaction with Mobile Devices and Services*, Amsterdam, the Netherlands, 2008. ACM.

[10] M. Kadous, and C. Sammut, "InCa: A Mobile Conversational Agent", *8th Pacific Rim International Conference on Artificial Intelligence*. Auckland, New Zealand, 2004.

[11] L.B. Larsen, K.L. Jensen, S. Larsen, and M. Rasmussen, "A Paradigm for Mobile Speech-Centric Services", *INTERSPEECH*, Antwerp, Belgium, 2007.

[12] M. Turunen, J. Hakulinen, A. Kainulainen, A. Melto, and T. Hurtig, "Design of a Rich Multimodal Interface for Mobile Spoken Route Guidance," *INTERSPEECH*, Antwerp, Belgium, 2007.

[13] J. Clawson, A. Voida, N. Patel, and K. Lyons, "Mobiphos: A Collocated-Synchronous Mobile Photo Sharing Application", *10th International Conference on Human-Computer Interaction with Mobile Devices and Services*, Amsterdam, the Netherlands, 2008.