Context-aware Dialog Strategies for Multimodal Mobile Dialog Systems

Jeongwoo Ko[1], Fumihiko Murase[2], Teruko Mitamura[1], Eric Nyberg[1], Masahiko Tateishi[2], Ichiro Akahori[2]

[1]Language Technologies Institute, School of Computer Science
Carnegie Mellon University, Pittsburgh PA 15213, USA
[2]Research Laboratories, DENSO CORPORATION
Nisshin-shi, Aichi-ken, Japan
jko@cs.cmu.edu, fmurase@rlab.denso.co.jp, teruko@cs.cmu.edu, ehn@cs.cmu.edu,
mtatei@rlab.denso.co.jp, iakahori@its.denso.co.jp

Abstract

Multimodal mobile dialog systems face new challenges which do not exist in traditional spoken dialog systems. A mobile dialog system should provide robust task management during network loss, and intelligent assistance in new environments. To provide these capabilities, a system must sense changes in the user environment, and communicate them effectively to the user. In the field of human-computer interaction, such systems are referred to as context-aware systems. Two possible approaches have been explored: in one, the user requests information when needed (information pull), and in the other, the system automatically offers relevant information (information push). In this paper, we present our dialog management strategy for context-awareness using an information push model. A preliminary evaluation conducted in both labaratory and driving environments shows that context-awareness significantly improved user performance.

Introduction

For the past few years, spoken dialog systems have been studied in pedestrian and automotive environments (Buhler *et al.,* 2002; Pieraccini *et al.,* 2003) to provide route guidance and remote data access (e.g. information on tourist sites, weather, restaurants).

Dialog systems for mobile environments are different from traditional spoken dialog systems for telephone and desktop environments (McTear, 2004). Mobile dialog systems do not have explicit dialog sessions like telephone-based systems and need more robust interactions to support small screens and noisy environments.

Another difference is that mobile dialog systems should support more dynamic user situations when the user moves to a location not previously visited. In addition, network connectivity is not stable (e.g. when the user goes through a tunnel or climbs a mountain), and the system should be robust enough to support dynamic changes in computing environments. In the field of human-computer interaction (HCI), *context-aware* systems have been developed to model the use of contextual changes in offering relevant information and services to the user (Schilit *et al.,* 1994; Dey and Abowd, 2000).

Figure 1 illustrates a dialog in an automotive environment. To provide immediate navigational guidance to the user, the system takes the initiative and interrupts the user, even if he or she is already talking to the system about something else. After delivering the next navigation instruction, the system continues the dialog by reminding the user of what he or she was talking about before the interruption.

U1: I want to go to Carnegie Mellon University.
S1: The distance to the destination is 100 miles.
It will take about two hours.
U2: I want to know the weather in Pittsburgh.

(The system breaks in with navigation instruction)

S2: To go to Carnegie Mellon University, please make a left turn at the next intersection.
S3: For what day do you want to know the weather in Pittsburgh?

Figure 1: Example of context-aware system-initiative dialogs (U: user utterance, S: system utterance)

This example shows that context-aware dialogs are important in intelligent mobile dialog systems. The SmartKom Mobile system (Malaka *et al.,* 2004) is another example. It adapts route guidance based on user preferences and notifies the user of an interesting sightseeing attraction when he or she moves to the vicinity of the site.

One issue in context-aware systems is how to communicate with the user when the system senses a contextual change. Two possible approaches have been explored: in one, the user requests information when needed (information pull), and in the other, the system automatically offers relevant information (information push). One recent user study compared these two approaches in a mobile tour guide domain to test if information push was acceptable to users (Cheverst *et al.,* 2002). The system pushed new context-relevant information onto the screen in response to contextual changes, and an evaluation indicated that participants

liked the push-based approach. However, when supporting information push in a spoken dialog system, context-aware dialogs should be carefully designed to minimize user distraction, which raises three interesting design questions for dialog management: what is good information, when is a good time to say it and how to say it to the user.

In this paper, we describe a dialog system that supports context-awareness while providing information to the user using an information push model, and present results from an empirical validation. We set up two systems in both laboratory and driving environments: one system supported context-aware features (context-aware system) and the other did not (context-unaware system). Twenty seven participants used two systems to finish their given tasks and completed a user questionnaire after the tasks. The objective assessment shows that the context-aware system improves user performance in terms of task duration and the number of user turns, while reducing the degree of user distraction in driving.

The remainder of this paper is organized as follows: Section 2 outlines the system architecture and Section 3 describes what contexts are used to provide context-aware dialogs. Section 4 explains a decision making process for estimating interruptability. Section 5 describes how the system adapts its screen and speech interfaces. Section 6 and 7 describe a preliminary experiment and its results, and Section 8 concludes the paper.

System Architecture

CAMMIA (Conversational Agent for Multilingual Mobile Information Access) is a multilingual dialog system designed to provide a combination of route guidance and mobile information access in English and Japanese (Nyberg *et al.,* 2002). CAMMIA supports multimodality in the form of a speech interface combined with a tactile screen. CAMMIA also provides context-aware information which is tailored to the current user (preferences, history) and current context (e.g., location, time-of-day). The latest version of CAMMIA was successfully demonstrated at the ASRU 2005 workshop (Ko *et al.,* 2005).

CAMMIA consists of four layers to support multimodal mobile information access: the user interface layer, the dialog management layer, the task management layer and the application layer (Ko *et al.,* 2006a). The user interface layer interacts with the user via a variety of input/output modalities. The dialog management layer consists of two subcomponents: the Dialog Manager and the Reasoner. The Dialog Manager maintains multiple ongoing conversations. Dialogs are represented as dialog scenarios: each scenario consists of a set of states and transitions between states. The Reasoner consists of i-Finder and i-Predictor to search for relevant information and estimate interruptibility. The task management layer explicitly maintains each information download request as a separate task object. When the network is not available, it monitors the network status and downloads the requested information after the network becomes available later. The application layer includes data that are required to service information requests from the Task Manager. Typical application layer components include relational database

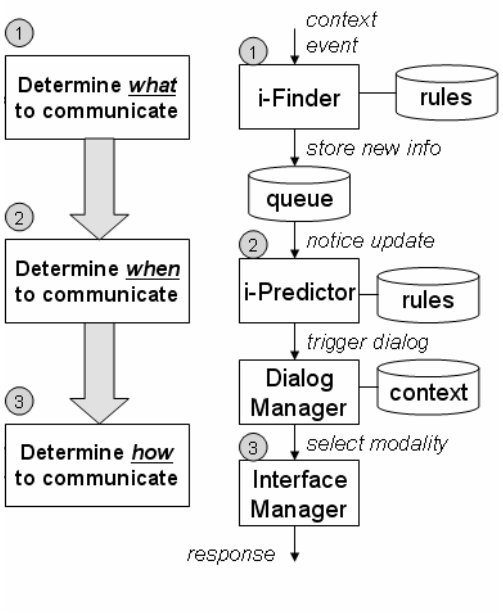

Figure 2: Framework for context-aware dialog management

adapters and web service client modules. Such components act as "wrappers" which provide information to the CAMMIA architecture from remote legacy services.

Context-aware dialog management

When the i-Finder notices any context change (e.g. navigational instruction, query results, etc.), the i-Finder decides whether this contextual change is useful to the user. If so, it stores the information in a pending information queue. The i-Predictor then accesses the pending information queue and determines whether it is a good time to interrupt the user. If the i-Predictor decides to provide the new information to the user, the information is sent to the Dialog Manager to trigger (or restart) the appropriate dialog with the user. If not, information is left on the pending information queue. After finding good timing, the Dialog Manager work together with the Interface Controller to adapt their behaviors for more efficient interactions and minimum user distraction.

User adaptation is an important aspect of the context-aware dialog management. CAMMIA maintains a user profile and allows the user to customize system services (e.g. features that are always checked, such as parking availability). CAMMIA also remembers the user's reactions to context-aware suggestions and updates the user profile accordingly. More intelligent user models and adaptation may be addressed in future work.

What to say

Context can be divided into four categories: computing context (e.g. network connectivity), user context (e.g. user location), physical context (e.g. traffic condition) and time

context (e.g. time of day) (Chen and Kotz, 2000). Table 1 lists the contextual events that trigger context-aware dialogs in CAMMIA.

Dialog	Contextual Event	Context Type
Navigational instruction	Navigation system sends route guidance	User context
Network-aware service	Network connectivity is changed	Computing context
Situation-aware confirmation	User sets a destination	Time context User context

Table 1: Contextual events and dialog types.

Navigational instruction is one of the most important contexts in mobile environments. When the navigation system asynchronously sends a navigation instruction to CAMMIA, it is added to the output queue and handled with a highest priority.

Network status change is another contextual event in user environments. When the network is not available (e.g. the car is inside a tunnel), the user can still use the dialog system in a limited way (e.g. to access cached data or to control in-car devices). In addition, CAMMIA preserves pending user requests into the information queue and monitors the network status. When the network becomes available again, the system downloads information for any pending requests and resumes the dialog (Figure 3); the user does not need to restart the dialog explicitly. If the network is unavailable for an extended period, the user may not remember that there are pending tasks, and the system first reestablishes the dialog context for the user. However, if the requested information is no longer relevant, depending on the context (e.g. elapsed time, location) the dialog is not restarted automatically. For example, the user may ask about a nearby restaurant for dinner. When the network becomes available, the user may be at a different place or it may be too late to have dinner there (e.g. 10 p.m.). In this case, it is better for the system to ignore the previous request. Table 2 shows two examples which decides whether to resume pending dialogs or not. The user can change the threshold in the user preferences or disable this feature if not needed.

Rule for the time: calculate the elapsed time between the current time and the request time if (elapsed-time $>$ TIME-THRESHOLD) then ignore
Rule for location: calculate the distance between the current location and the location the user asked about if (distance $>$ LOC-THRESHOLD) then ignore

Table 2: Example of domain specific rules.

User context and time context can be used for situation-aware confirmation. Before setting a destination or adding an intermediate destination to the navigation path, the system may check the availability of the selected destination. If the venue will be closed by the time the user is estimated to arrive, the system informs the user so that he or she can search for other places. In this case, the system takes the initiative to confirm a user action (e.g. movement to a selected location) based on reasoning with information that the user has not explicitly mentioned (opening times, availability of parking, payment options, etc.).

U1: What is the weather for the destination today?
S1: I am sorry, but the network is not available right now. I will let you know when the network has recovered. You may cancel the dialog if you don't want to wait.

(After 15 minutes, the network has recovered)

S2: You asked about the weather for the destination today. Do you want to hear about it?
(Display the following items on the screen:
Information which helps remind the user of the previous dialog such as requested time, requested location etc.
Options: yes, no, more information)
U2: Yes.
S3: The weather for the destination today is sunny and clear. (Display detailed weather information on the screen)

Figure 3: Example dialog of network-aware service (U: user utterance, S: system utterance)

When to say it

As the new information is provided under information push mode, it is important to find a good time to minimize user interruption. A Wizard of Oz experiment was conducted to explore sensor-based predictions of user interruptibility (Hudson *et al.,* 2003). The study showed that sensor-based statistical models such as a decision tree, SVMs, and naive Bayesian predictors could predict interruptibility with 75-80% accuracy. It also showed that simple sensors such as talking, using keyboard, using mouse and time of day obtained comparable results, 76.5% accuracy.

In our system, we have a manually created decision tree based on three sensors to reason about interruptibility. a) whether the user is talking, b) temporal proximity of direction change, and c) the road type. When the user is talking or if there is an active dialog, the i-Predictor decides that it is not a good time to interrupt the user. In this way, the system minimizes the possibility to overwrite the user's current dialog. The temporal proximity of directional guidance is used as another sensor. Right after or right before a directional guidance, the i-Predictor does not interrupt the user.

In automotive environments, a road type is important sensor. In the following subsection, we describe a preliminary experiment that focused on finding sensors that are useful in automotive environments.

Preliminary user study

Approach: Two field trials were done with different sub-

Road type	Dialogs started by the driver	Dialogs started by the wizard	Total dialogs
Highway	4	12	16
Local streets	0.5	5.5	6
Highway (return)	0.5	2.5	3

Table 3: Average number of dialogs per road type.

jects (two males in their 20's). The car drivers were accompanied by a human wizard equipped with a navigation system. While the subject drove, the wizard rode in the passenger seat and pretended to be a tour guide dialog system. The round trip journey from Nisshin City to Hamamatsu City (Japan) took about half a day by car. The route consisted of a highway, local streets, and another highway, a total about 160 km. The following task was given to the subjects upon departure: "Visit Hamamatsu city for sightseeing. The details of the trip, such as a restaurant for lunch, are not yet determined yet. Talk to the Tour Guide Dialog System on the way and decide driving plan accordingly."

Findings: Table 3 shows the average number of dialog topics for each road type. The topics include directions, traffic, weather, restaurants, sightseeing sites, and special regional food. As can be seen in Table 3, the drivers and the wizard tended to talk more frequently when on the highway towards the destination than on the return route. This was mostly because on the return route, the drivers were tired and the wizard did not mention topics that had already been talked about before. On local streets, they had fewer dialogs because the driving needed more attention. This supports the conclusion that the road type and trip phase are important features in driving environments.

The wizard sometimes introduced topics the driver could ask about when the driving time was long and the driver was silent for a long time, e.g., "If you feel sleepy, please let me know. I'll guide you to a rest area." The drivers remembered this and said, "I feel a little tired. Is there a rest area near here?" Providing suggestions about the system's functionality can be initiated based on the trip time and silence period.

Dialogs about tourist sites may follow different patterns, depending on whether they take place a) on the route to the destination, or b) close to the destination. We found that for case a), the driver wished to learn about nearby tourist sites but did not intend to visit them on the current trip. When the driver asks about tourist sites near or at the destination, he or she may be more likely to visit them. This shows that a sensor to check on whether the current location is the destination is important in a tour guide domain.

Based on the sensors acquired, we are planning to extend the use of sensor-based domain-specific rules. In addition, we are planning additional user studies for different situations, such as family trips and business trips. This may give us the training data to build statistical models and test how accurate such models are in estimating interruptibility.

How to say it

To minimize the effect of an interruption on the user and to support more efficient interactions, the system should be able to adapt its behaviors in certain environments.

Directional guidance has a highest priority and the system interrupts the user even though he or she is speaking (Figure 1). After delivering the next navigation instruction, the system continues the dialog by reminding the user of what he or she was talking about before the interruption.

For network-aware service, the system interacts with the user with two different dialog strategies. When the waiting period is short, the system directly presents the pending information without any negotiation dialog because the user probably remembers the context well enough to understand the relevance of the information. When the waiting time is long, the system reminds the user of what she was doing previously and provides options such as yes, no, or more information. The user can say "yes" if she wants to hear about it or say "no" if she does not need it any more. Or the user can ask more information if she does not remember the previous dialog. This makes the system display dialog history showing what she was talking about before network loss. This option may be useful if the user had a long conversation with the system.

When confirming a user choice that might violate a constraint, the system first finds alternate locations which satisfy the constraint. If only one alternative is found, the system informs the user by directly providing the alternative (e.g., "Luca restaurant will no longer be open when you arrive. But Pomodoro will be open."); three options are displayed on the screen: "Go Luca", "Go Pomodoro", and "More information about Pomodoro". When two or more alternative locations satisfy the constraints, the system informs the user (e.g., "Luca will no longer be open when you arrive. Do you want to search for other restaurants?"). On the screen two options are displayed: "Go there", and "Search other places".

Even though the system tries to estimate the best timing for an interruption, new information provided by the system may overwrite the current user context, e.g., by replacing what the user was viewing on the screen. In our multimodal dialog system, we added a back button on the screen interface, and added user utterances to the grammar whose corresponding speech act is to return to the previous context in the dialog.

Experimental Design

The experiment was conducted in both driving and laboratory environments. In the driving environment, the participants were asked to complete their tasks while maintaining a simulated speed of 100 km/h. A highway driving scenario was chosen because driving on a highway takes less effort than driving on local streets, and subjects might be over-

whelmed by attempting to use a dialog system while driving on local streets. To ensure high recognition accuracy, participants used a push-to-talk button near the steering wheel and a close-talk microphone. Since it is not safe for the driver to use a touch screen interface while the vehicle is moving, the participants were asked not to push buttons on the screen.

As a baseline test, the same experiments were done in the laboratory environment. This enabled us to estimate how much driving itself affects usage of a dialog system

Subjects

The experiments were conducted with native Japanese speakers. In the driving environment, the experiment included sixteen participants (four females and twelve males), ranging in age from 19 to 38. In the laboratory environment, the experiment included eleven participants (five females and six males) with ages ranging from 24 to 35.

Procedure

In the laboratory, the participants were trained for ten minutes. Training included microphone volume adjustment and an introduction to the system. In the driving environment, the participants were trained for fifteen minutes in order for them to become familiar with the driving simulator.

Each subject participated in three tasks for three types of context-aware features. Each task consisted of two subtasks: in one, participants used the context-aware system and in the other, they used the context-unaware system. After each task, subjects completed a user satisfaction questionnaire with a seven-point Likert scale. To minimize the within-subject learning effect, one half of the subjects started the tasks in the opposite order from the other half. Since the focus of the evaluation was to determine the utility and the feasibility of context-awareness in a spoken dialog system, the evaluation focused on just one of the features in each context-aware category where feature evaluation was feasible given our experimental setup. The next section describes the tasks in detail.

Tasks

Task 1 for Situation-aware confirmation. The participants had to find a nearby Italian/Japanese restaurant with a parking lot and select it as a trip destination. Some of the restaurants did not provide parking and some would not be open on arrival time. When the participant selected a restaurant that did not satisfy the search conditions, the system provided a notification and returned to the original search result state. The participant then continued to search for another restaurant. When using the context-aware system, the system notified the participant if the restaurant chosen did not have a parking lot or would be closed based on simple reasoning using restaurant business hours and the estimated time to the restaurant. When using the context-unaware system, the participants had to figure out parking lot availability and open status by themselves.

Task 2 for Network-aware service. The participants' task was to find out the weather in Tokyo for that day and the following day. To simulate network loss, we used a network loss simulator which made the network unavailable for 20 seconds. When using the context-unaware system, the participants had to monitor the network status and restart the dialog when the network became available. In the context-aware system, the system maintained pending user requests during network loss, and continued the dialog when the network became available and the context was valid (i.e. the participant was not talking to the system and there were no other active dialogs).

Results and analysis

Each set of test results was analyzed using both an objective assessment and a subjective assessment. For statistical significant tests, ANOVA (ANalysis Of VAriance between groups) was used. ANOVA tests the statistical significance of the differences among two or more groups whose size is different.

Objective assessments

Task completion: All the subjects successfully finished their tasks. The screen interface helped with task completion because the subjects were able to get hints about what they could say from the screen.

Time-to-completion: As can be seen in Table 4, it took less time using the context-aware system versus the context-unaware system to complete the required tasks. When comparing the data collected from driving and laboratory environments, more time was required in the driving environment for the user context and dialog context ($p<0.05$). The difference in the network context feature was not significant between the laboratory and driving environments, presumably because the task was simpler than the other tasks.

Turns-to-completion: The participants spoke much less when using the context-aware system than using the context-unaware system (Table 5). When comparing the data from the driving environment with the data from the laboratory environment, we see that in general the participants spoke a little bit more in the driving environment, but the difference was not statistically significant.

Variation in driving behavior: To measure the effects of the dialog systems on driving behavior, we compared the driving behavior under three conditions: driving without using a dialog system (baseline), driving while using the context-unaware system, and driving while using the context-aware system. In terms of average car speed, there was no significant difference among the groups. This indicates that the participants were able to maintain an average speed while using either dialog configuration. However, variance in car speed differed. When using the context-aware system, the variance in car speed was not significantly different from the driving-only baseline ($p=0.27$), but when using the context-unaware system the variance in car speed was greater than in the baseline test ($p<0.02$). As comparison on variance does not indicate how frequently the car speed changed, we also measured total speed changes. The results show that the total speed changes were less when using the context-aware system than using the context-unaware system ($p<0.03$). We also measured the angle between the car's direction and the center line on the road.

	Lab		Driving	
	unaware	aware	unaware	aware
Situation-aware confirmation	142 (±40)	103 (±30)	217 (±76)	130 (±34)
Network-aware service	42 (±11)	35 (±4)	46.4 (±8)	37.5 (±6)

Table 4: Average task completion time (unit: second)

	Lab		Driving	
	unaware	aware	unaware	aware
Situation-aware confirmation	18 (±4.4)	12 (±4.2)	20.6 (±7.4)	12.4 (±2.6)
Network-aware service	3.1 (±2.3)	1.2 (±0.6)	3.3 (±2.9)	1.6 (±1.5)

Table 5: Average number of user turns

The results show that there were not significant changes in driving angle when using a dialog system. But less change was noted when using the context-aware system ($p<0.05$). These results indicate that context-awareness could reduce the degree of user distraction during driving. More details on the analysis of driving behavior when using a dialog system can be found in (Ko *et al.,* 2006b).

Subjective assessments

The participants in both environments selected the context-aware system as easier to use and more helpful (for all cases, $p<0.001$). Overall satisfaction was lower in the driving environment than in the laboratory environment, presumably because the use of a dialog system while driving requires more attention.

CONCLUSION

In this paper, we described multimodal dialog management for context-aware dialogs under information push mode and addressed them in three different aspects: what to say, when to say and how to say it. When the user does not like information push mode, he or she can easily turn on/off context-aware features in user preferences or change options. How to easily add more useful rules and how to make them easy-to-use are one of future work.

We also described the experiment on context-aware dialog management. The experimental results showed that context-awareness significantly improved user performance in terms of time and the number of user turns. When comparing the driving environment with a laboratory environment, the participants tended to take more time to complete dialog tasks in the driving due to the effort required for the primary task (driving). But there was no significant difference in the number of user turns. Also the participants in both environments reported that context-awareness was easy to use, helpful for finishing tasks. An analysis of car speed and angle of travel shows that context-awareness may reduce the degree of user distraction during driving.

We are planning to run another experiment with a more complex driving environment, such as driving on local streets. Experiments which involve long-term usage and finding more useful contexts are another future work.

References

Buhler, D., Minker, W., Haubler, J. and Kruger,S. 2002. Flexible Multimodal Human-Machine Interaction in Mobile Environments. In *Proceedings of International Conference on Spoken Language Processing*.

Chen, G. and Kotz, D. 2000. A Survey of Context-aware Mobile Computing Research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College.

Cheverst, K., Mitchell, K. and Davies, N. 2002. Exploring Context-aware Information Push. *Personal and Ubiquitous Computing*, Vol 6 No 4 pp. 276-281.

Dey, A. K. and Abowd, G. D. 2000. Towards a Better Understanding of Context and Context-Awareness, *Workshop on The What, Who, Where, When, and How of Context-Awareness, at Conference on Human Factors in Computing Systems*.

Hudson, S., Fogarty, J., Atkeson, C., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J and Yang, J. 2003. Predicting human interruptibility with sensors: a Wizard of Oz feasibility study, In *Proceedings of Conference on Human Factors and Computing Systems*.

Ko, J., Murase, F., Mitamura, T., Nyberg, E., Tateishi, M., and Akahori, I. 2005. CAMMIA: A Context-Aware Spoken Dialog System For Mobile Environments, *Automatic Speech Recognition and Understanding Workshop, Demo Session*.

Ko, J., Murase, F., Mitamura, T., Nyberg, E., Hataoka, N., Sagawa, H., Obuchi, Y., Tateishi, M., and Akahori, I. 2006a. Robust Multimodal Dialog Management for Mobile Environment, In *Chapter 22 in Digital Signal Processing for In-Vehicle and Mobile Systems 2*, Abut, Hansen, and Takeda (Eds.), Springer Science, New York, NY, Scheduled for Spring 2006.

Ko, J., Murase, F., Mitamura, T., Nyberg, E., Tateishi, M., and Akahori, I. 2006b. Evaluating a context-aware spoken dialog system in mobile environments, *Conference on Language Resources and Evaluation*.

Malaka, R., Haeussler, J. and Aras, H. 2004 SmartKom mobile: intelligent ubiquitous user interaction. In *Proceedings of Conference on Intelligent User Interface*.

McTear, M. 2004 New Directions in Spoken Dialogue Technology for Pervasive Interfaces. In *Proceedings of*

Robust and Adaptive Information Processing for Mobile Speech Interfaces Workshop, COLING..

Nyberg E., Mitamura T., Placeway P., Duggan M. and Hataoka N. 2002. Dynamic Dialog Management with VoiceXML, In *Proceedings of Human Language Technology Conference*.

Pieraccini, R., Dayanidhi, K., Bloom, J., Dahan, J.G, Phillips, M., Goodman, B.R. and Prasad, K.V. 2003. A Multimodal Conversational Interface for a Concept Vehicle, In *Proceedings of Eurospeech*.

Schilit, B., Adams, N. and Want, R. 1994 Context-Aware Computing Applications, *IEEE Workshop on Mobile Computing Systems and Applications*.