Towards context awareness using Symbol Clustering Map

J. Himberg[1,2], J. A. Flanagan[1], and J. Mäntyjärvi[3]
(1) Nokia Research Center, P.O. Box 407, FIN-00045 NOKIA GROUP, Finland
Tel: +358 7180 08000, Email: adrian.flanagan@nokia.com
(2) Neural Networks Research Centre, Helsinki Univ. of Technology, Finland
(3) VTT Technical Research Centre of Finland

Keywords: symbol clustering map, mobile computing, context awareness

July 22, 2003

Abstract— Recognizing the context of use is important in making mobile devices simple to use. The device and the underlying mobile service can provide a personalized user interface that adapts to the usage situation. The device can infer parts of the context of the user from features extracted from on-board measurements of acceleration, noise level, luminosity, humidity, etc. In this paper we consider context recognition by fusing and clustering these context features using a recently introduced method, the Symbol Clustering Map. As such, it can be used for finding static patterns but a suitable transformation of the data allows identifying also temporal patterns.

1 Introduction

A common problem in data analysis is to combine information from various different and possibly noisy sources and to extract patterns from this data. A novel application, which requires combining diverse information sources is *context awareness* that has become a major topic in human-computer interaction [4, 11] also in mobile computing. In mobile communications, the usage situations, that is, the context, can vary a lot. On the other hand, a mobile terminal is often expected to enable connections all the time. At the same time, it should not irritate the user by signalling in a wrong way at the wrong moment, or by requiring constant attention to keep it working in the right way for the situation. In addition, the hand-held terminals are becoming more and more sophisticated in their function yet smaller in their size. The interaction could be made easier and less intruding if the mobile device recognized the user's current context and adapted its functions accordingly.

A widely cited definition in [4] states that context can be defined as any information that describes the relevant elements of a given situation. Thus, the first step in context recognition is to acquire information on the user and the environment. Information of user's preferences is obtained from the logs of different applications, e.g., calling, messaging, using calendar, or profiling. Piece of ambient information can be obtained by directly monitoring the user's physical environment using on-board sensors and information of user's location [1]. The operating network itself can offer information, e.g., on location of a phone. Setting explicit information sources, context tags, located in a short range network is another approach.

The second step comprises a fusion of these information sources and the extraction of the useful information needed to determine the user's context, i.e., the interpretation of the context [4]. When some basic set of features is fixed, one can extract relevant context information by using simple, statistical data analysis methods including clustering and time series segmentation [7, 9]. The recognized clusters/segments can then serve as "higher-level contexts" that show which combinations of the basic features form common patterns in the data. Other research efforts include context recognition experiments with Hidden Markov Models (HMMs) to determine the occurrence of audio-visual contextual events [2], or a combination of the Self-Organizing Map (SOM) [8] with Markov Chains [12].

In the third step, the recognized contexts and the user's expectations for appropriate actions in those contexts have to be combined in a satisfactory manner. Obviously, an immediate solution is a prespecified, rule based guidance of the user interface. For example, if the sensors detect that the user is running, the font used in the display can be made larger, or audio volume can be adjusted to compensate for higher levels of noise (see e.g., [10]).

A more advanced method would be that the device autonomously learned what actions/applications the user prefers in the recognized contexts and to suggest automating some of these. This is a challenge since it ultimately requires that a machine should react intelligently to everyday social situations. It requires sophisticated computing methods that can process information, preferably on-line, from a large and diverse number of sources. In addition, one must consider the limited power and computation resources of a mobile device. The efficient determination, combination and extraction of relevant information from diverse information sources is therefore a key issue in the area of mobile context awareness. Obviously, the context recognition system require also a well-organized soft-

ware and network infrastructure and consideration of privacy issues. Various aspects of context awareness and its implications have been covered in [4] and other articles of the same volume.

We narrow the scope in this paper only to the second stage of the process, recognizing the context. By combining different sources of context information a better, and potentially more useful, description of the context is obtained. Moreover, we use information only from on-board sensors; collecting this kind of data does not require any specialized tags of other infrastructure, only a device that contains a collection of sensors. We show how a recently developed algorithm called Symbol Clustering Map can be used successfully to infer user's context from this kind of data.

2 An approach to context recognition

In context recognition, as in any pattern recognition task, it is reasonable to fix a set of basic features that are extracted from the observed data. We present the features in symbolic form, for example, we say that ambient noise is "low", "medium", or "high" instead of presenting some real/integer value on a scale. This eases combining various kinds of information sources that might be accessible. For example, calendar markings or semantic descriptions of locations like "meeting" or "Helsinki" would be inherently symbolic. Moreover, in our work the basic features have been selected so that they reflect some everyday concepts. (see Sec. 4). This eases understanding the meaning of the extracted context and integrating rule based parts to the system.

In fact, we have fixed an alphabet $\Sigma_0 = \{S_1^0, S_2^0, \ldots S_N^0\}$ of context symbols. We refer to these context symbols as "context atoms" since the rest of the context inference emerges from these symbols. From this low-level of context representation, we wish to move on to a higher level of abstraction. By this we mean recognizing sets of context atoms that seem to form frequently occurring patterns in the context atom stream. We say that a higher-level context can be obtained from

- the simultaneous fusion of context information from several lower-level context sources, e.g., "running" and "railway station" might imply "being late", or
- sequential fusion of one lower-level context information source into a higher-level context, e.g., "browsing bus time table","walking-inside" succeeded by "walking-daylight" might imply "heading for a bus stop"

The included examples are only illustrative. The inference that is made here cannot make up the semantic meanings, that is, the names of the contexts. Those could be provided by an expert, e.g., the user—but semantic labelling of the contexts is actually not needed if the aim is to associate actions to contexts by learning from examples. Next, we consider these two basic ways of fusing information in more detail:

Multi-source fusion We combine the information that is available at time instant t on level i in alphabet Σ_i in the context hierarchy. A higher-level context is generated and represented by the fusion of the n context states available at time instant t. We can characterize each moment t by giving the set of n context atoms that happen to be active at that moment and labelling this as a new context symbol on a higher-level alphabet Σ_{i+1}

$$S^{i+1}(t) = \{S_1^i(t), S_2^i(t), \ldots, S_n^i(t)\}$$

where $S_k^i(t), k = 1, \ldots, n$ are some symbols of the lower-level alphabet Σ_i. Alphabet Σ_1 is obtained by combining the symbols on the lowest level, i.e., the context atoms. In practice, the same context does not always produce exactly the same context atoms either due to variations in the environment or simply by the imperfection of feature extraction. These variations in context atom sets can be considered as noise. Finding the prototypical context atom sets corresponding to the contexts given this noisy data is obviously a clustering problem.

Temporal fusion Another way of forming the higher-level context is to combine context symbols sequentially in time in a string:

$$S^{i+1}(t) = (S^i(t-k), S^i(t-k+1), \ldots, S^i(t))$$

This differs from the previous way in the sense that now the order of the context symbols is of interest. This is a problem of finding event episodes [3].

Both forms of generalization can be combined in a hierarchical manner and mixed. For example, in the experimental section, a set of context atoms Σ_0 is given, we form second stage context symbols Σ_1 by multi-source fusion, and then yet another stage Σ_2 using temporal fusion.

Standard statistical clustering and segmentation methods that work on real vector space and/or process the data in batch have the potential to perform both the sensor fusion [9] and the temporal fusion [7]. However, due to the symbolic nature of the data and the computational limits, we use here a method for automatic recognition of noisy symbol sets, the Symbol Clustering Map (SCM) [5]. The method is implemented in a serial fashion that learns clusters in the input data and automatically assigns labels to the clusters without user intervention. The algorithm is described briefly in the following section.

3 Symbol Clustering Map

The SCM is based on a lattice structure, generally 2 dimensional with a total of $M \times M$ lattice points. Associated with each node k of the lattice is a symbol set $\mathbf{S}_k(t) = \{s_1(t), s_2(t) \ldots, s_{n(k,t)}(t)\}$, from now on referred to as the "node set" and associated with the node set is a weight vector $\mathbf{X}_k(t) = (x_1(t), x_2(t), \ldots, x_{n(k,t)}(t))$, where $x_j(t) \in [0,1]$, $\forall\ j,t$ and $n(k,t)$ is the number of symbols in the node set $\mathbf{S}_k(t)$, the same as the number of weights in the weight vector $\mathbf{X}_k(t)$. Note that each $x_j(t)$ of $\mathbf{X}_k(t)$ is directly associated with the symbol $s_j(t)$ of $\mathbf{S}_k(t)$. Initially the weight vectors $\mathbf{X}_k$ and node sets $\mathbf{S}_k$ are set to random values with $x_j(0) \in [0,1], \forall\ k,j$. At time step t there is an input symbol set $\hat{\mathbf{S}}(t) = \{\hat{s}_1(t), \hat{s}_1(t), \ldots, \hat{s}_{\hat{n}(t)}(t)\}$. For this given input the similarity between $\hat{\mathbf{S}}(t)$ and each of the nodes k, is calculated and given by an activation function $\mathcal{A}_k(t)$. The activation $\mathcal{A}_k(t)$ is defined as,

$$\mathcal{A}_k(t) = \frac{\left(\sum_{i=1}^{\hat{n}(t)} \sum_{j=1}^{n(k,t)} \hat{\delta}_{k,i}(t)\delta_{k,j}(t)\right)^2}{n(k,t)\ \hat{n}(k,t)} \qquad (1)$$

and for $s_j(t) \in \mathbf{S}_k(t)$, the $\delta_{k,j}(t)$ function is defined as,

$$\delta_{k,j}(t) \ = \ \begin{cases} 1 & \text{if } s_j(t) \in \mathbf{S}_k(t) \\ 0 & \text{if } s_j(t) \notin \mathbf{S}_k(t) \end{cases}$$

and $\hat{\delta}_{k,i}$ is defined in a similar manner. The value of the activation function is maximal, if the node set and the input symbol set are identical and decreases towards zero the less there are symbols in common for the node set and the input symbol set. The winner node $v(t)$ at time t is chosen as the one with maximum activation, hence

$$v(t) \ = \ \underset{1 \le k \le M \times M}{\arg\max}\ \mathcal{A}_k(t) \qquad (2)$$

This represents the first step of the clustering algorithm, deciding on the winner node that best represents the input.

The second stage is the updating of the nodes. For each node $k = 1, 2, \ldots, M \times M$ the update rules can be briefly explained as follows. In the update rules there is a neighborhood function $h_m(v(t), k)$, a Mexican hat as a function of the lattice distance d_L between the winner $v(t)$ and the node k being updated. The update rule consists of four parts: *I)* if the symbol $s_j(t)$ is present in the input $\hat{\mathbf{S}}(t)$ and the value of the Mexican hat function is positive, the weight component $x_j(t)$ is increased towards 1. Otherwise, if the Mexican hat is negative the $x_j(t)$ is driven towards 0. The first part corresponds to a reinforcement learning while the second is an inhibition. *II)* if the symbol $s_j(t)$ is not present in the input $\hat{\mathbf{S}}(t)$ then the weight $x_j(t)$ is driven towards 0 irrespective of whether the Mexican hat function is positive or negative. This can be considered a form of unlearning. *III)* if there is an element $\hat{s}_j(t)$ of the input that is not in $\mathbf{S}_k(t)$ then it is added into $\mathbf{S}_k(t)$ and given an initial weight value proportional to $\alpha(t)$ and $h_m(v(t), k)$. If $h_m(v(t), k) \le 0$ then the symbol is not added. *IV)* if a weight is smaller than a threshold β, i.e., $x_j(t) < \beta(t)$, $\beta(t) \in [0,1]$, the weight and the corresponding symbol are removed from the node set and weight vector.

Intuitively, what the algorithm does is to determine patterns of frequently occurring symbols in the input. The weights of the weight vector associated with a node set indicate the level of probability of the symbol occurring in the symbol set. The cutoff function $\beta(t)$ increases in time, e.g., from $0.1 \rightarrow 0.3$ and removes less frequently occurring symbols from the node set. In this sense, the node sets represent average values of the input sets. This means that the most frequently occurring symbol sets occur at the cluster centers of the input.

This learning algorithm cannot be used as such for identifying temporal patterns since the similarity measure only works for unordered symbols sets However, a simple transformation of the symbol sequence can be used to circumvent the problem. Assuming that the symbols S of alphabet Σ are integers, an n-tuple, say the triplet $(S(t-2), S(t-1), S(t))$ is coded into a single symbol, $\bar{S}(t) = S(t-2) * N^2 + S(t-1) * N + S(t)$ where $N = |\Sigma|$. It is obvious that every triplet has a unique coding. This coding is carried out for every symbol $S(t)$ in the sequence resulting in a set of symbols where the ordering can be considered unimportant but which still maintains some of the information of the ordering in the original sequence.

4 Experimental results

The experiments in this section are the same as in [6]. The same data set has been used also in [7, 9].

4.1 Test setup and context atoms

A mobile phone was equipped with a set of sensors that included sensors for ambient illumination, noise, air humidity, and temperature, as well as a galvanometer for sensing touch and a three-axis accelerometer. Two user's then went through one of the 5 scenarios described in Tab. 1, and each scenario was repeated successfully about 25 times for two test persons. Each repetition of a scenario lasted about 3–4 minutes depending on the testee, the scenario and the environment. When the terminal was not on the table it was hanging in front, from the user's neck.

The signals were recorded and various feature extraction algorithms were applied in order to generate context atoms. The context atoms were collected in groups that describe orientation, stability, touch, frequency of local AC current (estimated from the spectrum of the ambient light), illumination level, temper-

ature, humidity, ambient sound level, and type of walking. Each context atom was assigned a symbol, in this case an integer value. A "dummy" context atom × was assigned if it was not possible to select any context atom in a group. The context atoms in each group are: *Orientation*: 1=Display Down, 2=Display Up, 3=Antenna Down, 4=Antenna Up, 5=Sideways Right, 6=Sideways Left, 7=×; *Stability*: 8=Stable, 9=Unstable, 10=×; *Touch*: 11=In hand, 12=Not in hand, 13=×; *AC frequency*: 14=50 Hz, 15=60 Hz, 16=×; *Illumination level*: 17=Bright, 18=Modest, 19=Dim, 20=Dark 21=×; *Light source*: 22=Natural, 23=Artificial; *Temperature*: 24=Hot, 25=Warm, 26=Cool, 27=Cold, 28=×; *Humidity*: 29=Humid, 30=Normal, 31=Dry, 32=×; *Sound level*: 33=Silent, 34=Modest, 35=Loud, 36=×; *Walking*: 37=Slow, 38=Fast, 39=Running, 40=×. During the experiment, and once every second, a set of context atoms was produced. For example, $\{4, 9, 12, 14, 18, 23, 25, 29, 33, 37\}$ could be interpreted as "antenna up, unstable, not in hand in artificial light at 50 Hz" etc.

4.2 Context recognition

The context atoms in Sec. 4.1 represent the first level of context. The second level of context is generated by combining the context atoms into a symbol set as described in the previous section, and then recognizing each symbol set as a particular context using the SCM.

During the repetitions of the scenarios, the user spends much more time on some activities than others, e.g., a lot of time is for walking, but little time for when the terminal is in hand. In order to avoid the rare contexts to disappear in clustering, each repetition was adaptively downsampled as follows: Each repetition was segmented into 14 subsequent segments using a simple algorithm that detects and inserts a segment when there is a significant change in one or several values of the context atoms. One sample symbol set from each segment was randomly chosen as a representative of the segment.

With a total of 241 repetitions of the 5 scenarios, and 14 samples for each scenario, a total of $3374 = 241 \times 14$ symbol sets were used in the off-line training of the SCM. The training was carried out by randomly selecting one of the 3374 symbol sets as input to the SCM. Fig. 1 shows the symbol set associated with nodes of the SCM lattice and Fig. 2 the associated weight vectors after 10000 iterations.

This is now the result of multi-source fusion and we call it SCM1. There are five clearly distinguishable clusters separated by nodes with null symbol sets and weight vectors. As the clusters are completely separated by null nodes, finding and labelling the clusters is a simple task. In Fig. 1, the clusters are labelled 1–5. For example, the cluster 3 with $\{4, 9, 12, 16, 17, 22, 25, 29, 33, 38\}$, indicates the user's

Table 1: Description of the scenarios. Activity "start": the testee takes the phone laying on the table, stands up and walks out of the room; "stop": the same actions but in opposite order. *Emphasized*: activity takes place outside, otherwise inside.

Activity	Location
Scenario 1, 44 repetitions	
start	office
walking	corridor
walking	stairs
walking	lobby
walking	*street*
walking	lobby
walking	corridor
walking	stairs
stop	office
Scenario 2, 48 repetitions	
start	office
walking	corridor
walking	stairs
halt	mail lockers
walking	*yard*
halt	mail lockers
walking	stairs
walking	corridor
stop	office
Scenario 3, 49 repetitions	
start	office
walking	corridor
halt	lift
walking	corridor
halt	*balcony*
walking	corridor
halt	lift
walking	corridor
stop	office
Scenario 4, 50 recordings	
start	office
walking	corridor
sitting+talking	meeting room
walking+talking	corridor
sitting+talking	coffee room
walking	corridor
stop	office
Scenario 5, 50 recordings	
start	office
walking	corridor
halt	lift
walking	corridor
halt	lift
walking	corridor
stop	office

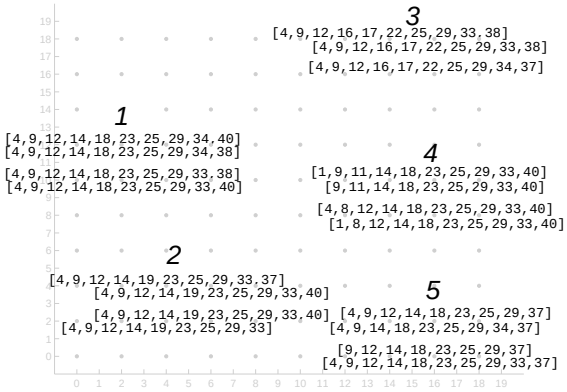

Figure 1: The node sets of the SCM1 lattice and the cluster labels. As in Fig. 2, only every second symbol set and weight set is plotted for ease of illustration.

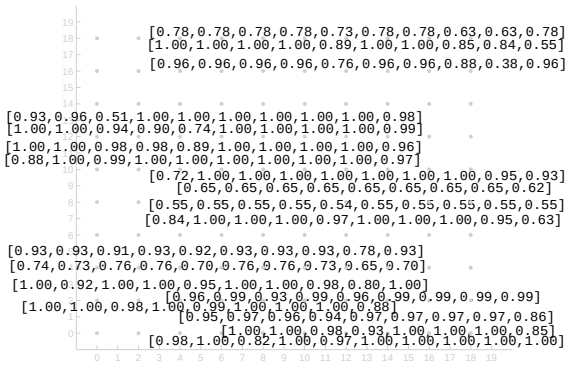

Figure 2: The weights corresponding to the node sets in Fig. 1

action, such that the phone antenna is up (4) and it is not hand (12) and unstable (9). Hence, the user is most likely moving with the terminal hanging in front. Furthermore the light is natural and bright (16, 17, 22) and it is warm and humid with a low sound level (25, 29, 33) and finally the user is walking fast (38). Obviously this cluster corresponds to a context where the user is walking outside. However, included in the same cluster is node sets containing 37 indicating that the cluster corresponds to any form of walking slow or fast. All other clusters have reasonable and most importantly, different, interpretations, that is, each cluster corresponds to a different context. Now, we add a second level of context recognition and consider describing one repetition of a scenario. We recode the symbol set at each time sample by the cluster label of SCM1; the label is that of the cluster to which the winning node for the symbol set at a time t belongs. This results in a symbol sequence, e.g., $(3, 3, 1, 1, 4, 5, 4, \ldots)$. The symbol sequence is further reduced so that adjacent repeating symbols are only listed once, e.g., the previous symbol string is reduced to, $(3, 1, 4, 5, 4, \ldots)$. In this, case the temporal relations are important and, therefore, the transformation explained in the end of Sec. 3 was used. Now, the SCM algorithm could be reapplied to these new symbol sets describing each repetition of the scenario. The repetitions were classified to a particular cluster, and we call this result SCM2. We will not reproduce the node labels and weights of SCM2 here because of limited space but Tab. 2 gives a confusion matrix between the known label of scenario 1–5 and the four cluster that SCM2 assigns to the scenarios.. The results suggest that scenarios S1, S2 are quite similar with little similarity between S1–2 (joined), S3, S4, and S5. From the description of the scenarios in Tab. 1 one sees that S1, S2, S3 all have in common that the user goes outside for some time, while in S4 and S5 the user is constantly indoors. Scenarios S1 and S2 are indeed quite similar but different from S3 since in S1–2 the user walks outside, whereas in S3 the user is moving about on a small balcony. From scenarios S4 and S5 it is seen from Tab. 1 that in S4 there is talking while in S5 there is none. There is some confusion between S4 and S5, however, this is due to the fact that there is no context atom for identifying speech but only for loudness of the ambient sound. Now, the second clustering (SCM2) presents a set of contexts of higher level than the first clustering (SCM1) which in turn is a higher level representation for the original context atoms.

Table 2: The number of the repetitions of each of the scenarios S1–S5 assigned to each cluster C1–C4 in the second clustering stage

	S1	S2	S3	S4	S5
C1	43	47	4	0	0
C2	1	1	45	0	0
C3	0	0	0	43	4
C4	0	0	0	7	46

Figure 3: The beginning of a repetition of scenario 1 with illustration of a context aware UI.

Figs. 3 and 4 illustrate a simple context aware application that changes the profile of the phone according to the context recognized by SCM1. Fig. 3 shows a video snapshot recorded from one of the repetitions of scenario 1 and the user interface of the phone. The user

Figure 4: The user is walking outdoors in scenario 1.

sits in office, and SCM1 cluster this state to cluster 4 (Fig. 1) and we have associated a "working profile" to be set automatically for this context. Fig. 4 is from a later phase of the same repetition where the user is walking outside. SCM1 recognizes this as cluster 3 (Fig. 1) and we have associated two actions, "key lock" and "outdoors profile", for this context.

5 Conclusion

The problem of context recognition has been shown to provide an ideal application for the use of data mining algorithms. By using a symbolic representation of context and combining and generating context based on symbol sets, context recognition becomes a problem of clustering. It has been shown how data from very different sources can be fused together using a symbol set representation and Symbol Clustering Map (SCM) algorithm. Processing the data using SCM does not require any user intervention or storage of large amounts of data. This makes the algorithm especially suitable for mobile computing. SCM has been applied to real world context data originating from different usage scenarios of a mobile device. The context data is produced by an experimental mobile terminal capable of monitoring its movements and environment by various sensors. It has been possible to classify different repetitions of the usage scenarios using SCM on the context data. It seems that the level of information available and the differences between the scenarios was not big enough to completely separate the different scenarios, but the principle of generating higher-level context and recognizing context has been demonstrated using a real, noisy data set and unsupervised clustering of symbol sets.

References

[1] P.J. Brown, J.D. Bovey, and X. Chen. Context-aware applications: from the laboratory to the market place. *IEEE Personal Communications*, 4(5):58–64, 1997.

[2] B. Clarkson and A. Pentland. Unsupervised Clustering of Ambulatory Audio and Video. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing 1999*, volume 6, pages 3037–3040, 1999.

[3] P. Smyth D. Hand, H. Mannila. *Principles of Data Mining*. MIT Press, Cambridge, Massachusetts, London, England, 2001.

[4] A.K. Dey, G.D. Abowd, and D. Salber. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction*, 16(2–4):97–166, 2001.

[5] J.A. Flanagan. Unsupervised clustering of symbol strings. In *2003 International Joint Conference on Neural Networks (IJCNN 2003)*, Portland, Oregon, USA, 2003. to appear.

[6] J.A. Flanagan, J. Mäntyärvi, and J. Himberg. Unsupervised clustering of symbol strings and context recognition. In *Proc. of the 2002 IEEE Conference on Data Mining (ICDM2002)*, pages 171–178, Maebashi, Japan, Dec 2002.

[7] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmäki, and H.T.T. Toivonen. Time Series Segmentation for Context Recognition in Mobile Devices. In *Proc. of the 2001 IEEE Conference on Data Mining (ICDM2001)*, pages 203–210, San José, CA, Dec 2001.

[8] Teuvo Kohonen. *Self-Organizing Maps*. Springer, Berlin, Heidelberg, 2001. (Third Extended Edition).

[9] J. Mäntyjärvi, J. Himberg, P. Korpipää, and H. Mannila. Extracting the Context of a Mobile Device User. In *Proc. of 8th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine System*, 2001.

[10] J. Mäntyjärvi and T. Seppänen. Adapting Applications in Mobile Terminals Using Fuzzy Context Information. In *Proc. of the 4th Int. Symp. on Mobile Human-Computer Interaction (Mobile HCI 2002)*, volume 2411 of *LNCS*, pages 95–107, Pisa, Italy, Sep 2002. Springer.

[11] B. Schilit, N. Adams, and R. Want. Context-Aware Computing Applications. In *Proc. of the Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, Dec 1994. IEEE Computer Society.

[12] A. Schmidt, K.A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, and W. Van de Velde. Advanced Interaction in Context. In *Hand Held and Ubiquitous Computing*, number 1707 in Lecture Notes in Computer Science, pages 89–101. Springer-Verlag, 1999.