Context aware user notification services

CHAPTER 1

INTRODUCTION

Context: Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

Use of Context: More recently, context-aware computing has emerged as a complementary approach, promising to reduce users' participation in carrying out a computation task. The approach strives to integrate computers into the environment, rather than taking them as distinct objects, so that they can be able to seamlessly gather and employ implicit surrounding information in order to establish a shared understanding. The reason for establishing a shared understanding has been clearly expressed by Weiser as follows:

The idea first arose from contemplating the place of today's computer in actual activities of everyday life. In particular, anthropological studies of work life teach us that people primarily work in a world of shared situations and unexamined technological skills. For example, when people attend a meeting their eyes communicate to convey agreements or disagreements to what is said or unsaid; voices are whispered to exchange impromptu opinions; facial expressions reveal to the other participants fatigue, boredom, or disinterest. More importantly, speeches may not be grammatically correct or even complete. Previous as well as unfolding incidents enable the audience to capture what cannot be expressed verbally. Speakers shift from one language to another and use words with multiple meanings, and still the other participants can follow.

Understanding Context: In the example of attending a meeting, it has been assumed that the context encompassing the interaction between participants is effortlessly recognized by all participants. Consequently, within this context, many activities are carried out, some of which are certainly unpremeditated activities yet consistent with the context. Some of the activities express the freedom (or flexibility) associated with the

recognition of the context – for example, using incomplete or incorrect statements, or using words with multiple meanings. Other activities reflect the participants' adjustment of behaviour in compliance with the context of interaction – for example, participants whispering to exchange impromptu ideas. The additional assumption has been that a context is considered to be distinct from activities, and that it describes features of the environment where in activities take place. In reality, however, the distinction between a context and an activity is not always clear. The reason for this is that the scope and usefulness of a context is limited to a particular setting, particular instances of action, and particular parties to that action [4]. For example, among the set of activities enumerated earlier, some or all of them can be considered to be contexts to other activities affected by the occurrence of these activities. A meeting by itself is an activity triggered by the occurrence of a context preceding it. Therefore, building a shared understanding requires the understanding of a context as a dynamic construct the relevance of which is determined by episodes of use, social interaction, internal goals, and local influences [3].

Definiton Of Context : Since a context plays a central role for a computing device to establish a shared understanding of the situation in which it operates, understanding the meaning of a context is an essential step.

Schilit defines a context to be the state of a constantly changing execution environment . An executing environment encompasses the user's condition (such as the user's location and social status), the physical environment (noise, lighting, etc), and the computing environment (availability and capability of computing devices as well as communication infrastructures).

Pascoe [1] describe context as the subset of physical and conceptual states of interest to a particular entity, this includes: the user's location, identity, physical environment and time.

According to Dey, if a piece of information can be used to characterize the situation of a participant in an interaction, then that information is a context [5]. Dey recognizes activity, identity, location and time as more important contexts than others [6].

More recently, Wang [7] identify three classes of real-world objects (user, location, and computing entities) and one class of conceptual object (activity) to characterize smart spaces. According to the definitions above, a context is viewed as information which is

discernable, predictable, and stable. As information, it can be known in advance, encoded, and represented just as other information is encoded and represented in software systems [5].

Consequently, it is possible for the designer of a context-aware application to (1) enumerate the set of contextual states that may exist, (2) know what information could accurately determine a contextual state within that set, and (3) state what appropriate action should be taken given a particular state [8]. In a recent paper, Dourish argues that such a view reflects a misunderstanding of the nature and role of contextuality in actual every day affair, "since contextuality comes about only when it is mutually recognized by all the parties to some interaction, drawing on their everyday, cultural, and commonsense understanding of the nature of the social events" [6]. According to Dourish, a context should not be understood as something that describes a setting; instead, it should be understood as: A relational property that holds between entities or activities. In other words, what could be viewed by some as a context may not be viewed by others as a context. This refers to the relative nature of a context. Something the scope of which is defined dynamically. A context does not have as such predefined states or set of propositions. The various features of a context should be defined dynamically. An occasional property. This refers to the temporal nature of a context, i.e., a context is particular to each occasion of activity and action. Something that arises from the activity. That means, a context is produced, maintained and enacted in the course of the activity at hand.

Features Of Context-Aware System: Pascoe proposes three basic features by which a context-aware system can be characterised, namely, contextual sensing, contextual resource discovery, and context adaptation [1]. Context sensing refers to the ability of a system to augment the sensing capacity of a user by capturing a relevant context, so that an associated action can be performed by the system. Contextual resource discovery refers to the ability of a system to search and bind to resources, and context adaptation refers to the adjustment in behaviour of a system in accordance with the social as well as physical settings wherein a computing task is carried out. To design and implement such a system, Dey proposes a framework with a set of required features. These features are: context specification, separation of concern and context handling, context interpretation, transparent distributed communications, constant availability of context acquisition, context storage, and resource discovery [2]. Context specification refers to specifying the system's behaviour and the types of contexts required for the behaviour. Separation of concern and context handling refers to the separation of context acquisition from context usage. Context interpretation refers to the provision of an appropriate abstraction to a piece of context so that it is meaningful to the system that uses (consumes) it. Transparent distributed communications refers to establishing a mechanism for acquiring a context in a transparent manner from a source which is not directly connected to the device on which a context-aware system is running. Constant availability of context acquisition refers to the constant availability of sources which deliver the contexts that are identified by the developer at the time the system is designed. Context storage refers to the persisting of contextual data for future reference. Finally, resource discovery, refers to the ability of a context-aware system to search and bind to context sources at runtime. Some of the features of the framework of Dey are prohibitive in not allowing a system to capture dynamic, real-world situations for the following reasons: Specifying the behaviour of a context-aware system at design time prohibits a user from defining a user-specific behaviour which might not been foreseen by the developer. Determining at design time a context which causes certain behaviour to occur

restricts a context-aware system from learning a new type of context which may equally cause the same behaviour to occur. Since a context-aware behaviour and the contexts associated with the behaviour can be unforeseen at design time, determining specific hardware and sensors for capturing a context may not be feasible. Furthermore, in a dynamic computing environment where the state of available resources and the resources themselves change over time, requiring a constant availability of sources for capturing specific context types can be unrealistic.

Context-aware: 'A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task. Pervasive Computing is the vision of technology that is invisibly embedded in our natural surroundings. Users are offered unobtrusive services that require minimal attention.The main aim of this project deals with a context aware service platform which can adequately relate user's context and services and automatically generates context aware services based on the user's habits. Context awareness is the capability of networking applications to be aware of the existence and characteristics of the user activities and environments. System have to adapt their behavior based on the current conditions and the environment they are immersed in. Such information commonly referred to as 'context'. The main aim of this project deals with a context aware service platform which can adequately relate user's context and services and automatically generates context aware services based on the user's habits.. The context awareness platform can predict the most relevant services that users will use in current situation based on their habits, and provide services in different modes. As learning and predicting mechanisms are on the basis of a stochastic approach Hidden Markov Model can absorb various uncertainties arising from sensor data and provide truly personalized services.

Notification Services: Context awareness [1] has emerged as a key element in pervasive computing. It facilitates applications that are aware of their environment and enables them to adapt to the current context. A key application that can benefit from context aware is a notification service. With context awareness notification service is able to keep track of changes in environment and can appropriately communicate these changes to applications that users are currently using. In this project we propose context aware user notification services which is based on the HMM and provides notifications

depending on the user's context. If users want to receive personalized notifications, they create individual user notification profiles. A user notification profile defines which level of intensity is appropriate in which context. The intensity of notifications can be based on availability of the user that is to be notified, or where the user s currently located. A further option is to take the co-presence of other persons into account. context-aware services based on the users' habits is necessary. We are developing a context-aware service platform called Synapse, which can learn different users' habits by exploiting the recorded histories of contexts and services, and then predict and provide the most relevant services that users will use in current situation based on their habits. We faced several challenges when we designed our system. First, considering the flexibility of system and the ease of management for end-users, we should apply a dynamic mechanism rather than binding the contexts and services in a specification language such as ECA [7]. Second, since users' habits may slowly change as time advances, our algorithms should have the ability of updating to reflect it. Third, corresponding to the diverse contexts (such as "the user is sitting", "the brightness in a room") and various services (such as "turn on light", "select TV channel 3"), our model should have the capability to deal with multi-dimensional inputs and outputs. Fourth, personalized services are desired by different users. Finally, the system should work with imperfect and noisy sensor data. With these challenges in mind, we apply a stochastic approach for Synapse, which is based on one of Bayesian Networks [7] , HMM (Hidden Markov Model) [8]. The model of Synapse consists of continuous cycles. Each cycle is composed of two phases: Learning Phase and Executing Phase. In the Learning Phase, Synapse learns the relationship between contexts (we call them "sensor events" in Synapse) and services by exploiting the recorded histories of them. Then in the Executing Phase, based on the learned relationship and the current sensor events, Synapse predicts the most possible services to be used and provides them to users. Since users would like to enjoy autonomous services in a moderate degree without losing control of them [2], Synapse provides services in two modes: Active Mode will start a service automatically based on sensor events, while Passive Mode recommends the top 5 relevant services in a list and let users select. The results of the Learning Phase are used as prior knowledge for the next cycle. To easily achieve personalization, user ID is treated as a sensor event. The

related works of time-series prediction and smart home projects will be introduced in section 2. The architecture of Synapse will be explained in section 3. The preliminary evaluation of Synapse will be discussed in section 4. The conclusion and future work will be given in last section.

RELATED WORKS:

For time-series prediction of continuous data, linear models (such as ARIMA, ARMAX [5]) or non-linear models (such as neural networks or decision trees [4]) are usually used. For discrete data, n-gram models or variable-length Markov models are common choices. Compared to these methods, Dynamic Bayesian Networks (DBN) [7] have some advantages appropriate for the challenges we face: First, it is easier for DBN to deal with multi-dimensional inputs and outputs. Second, prior knowledge is easy to be incorporated, so the prediction of the future is based on all the past history. Third, DBN is more flexible than simple supervised classifiers. Finally, DBN has been successfully used in many areas [6] for time-series prediction. Therefore, we choose HMM [8], one of Bayesian Networks, to build the core model of Synapse. This core model is a general context-aware platform, which can be used not only in smart home environment, but also in a broad range of context-aware applications that need to correlate the contexts and services, since it provides standard interfaces for contexts and services. Several smart home projects are in progress. The Georgia Tech Aware Home [1] and MIT Hussein [7] use an array of sensors to determine users' locations and activities within an actual house. The Neural Network House [6] balances the goals of anticipating user needs and energy conservation through a neural network. The MavHome [3] uses an intelligent and versatile home agent to perceive the state of the home through sensors and act on the environment through effectors. The industrial examples are also available, such as the Microsoft Easy Living project, the Cisco Internet Home, and the Verizon Connected Family project. Although, similar with these projects, our smart home test bed of Synapse extracts contexts from raw sensor data and adopts services from smart devices, our original core model guarantees the uniqueness of Synapse.

CHAPTER 2

LITERATURE SURVEY:

Hidden Markov Model: A Markov chain or process is a sequence of events (called states) the probability of each of which is wholly dependent on the event immediately preceeding it. Given a sequence of Q = $\{q_1,q_2,q_3\ldots,q_n\}$ the state of q is determined as:

$P(q_n|q_{n-1},\ldots..,q_2,q_1) \approx P(q_n|q_{n-1})$.

A Hidden Markov Model (HMM) represents stochastic sequences as Markov chains; the states are not directly observed, but are associated with observable evidences, called emissions, and their occurrence probabilities depend on the hidden states [6]. The generation of a random sequence is the result of a random transition in the chain. HMM has been applied to a wide variety of dynamic systems, the most salient applications being the ones dealing with speech recognition. For these applications, the hidden states are the smallest units of a speech called phonemes. Every word is thus built from phonemes, which are identified with the hidden states. After different hidden Markov models are trained on examples, one can run each HMM separately on a new word to be recognised. Then the likelihood of every HMM is computed on the new word and the highest likelihood is chosen.

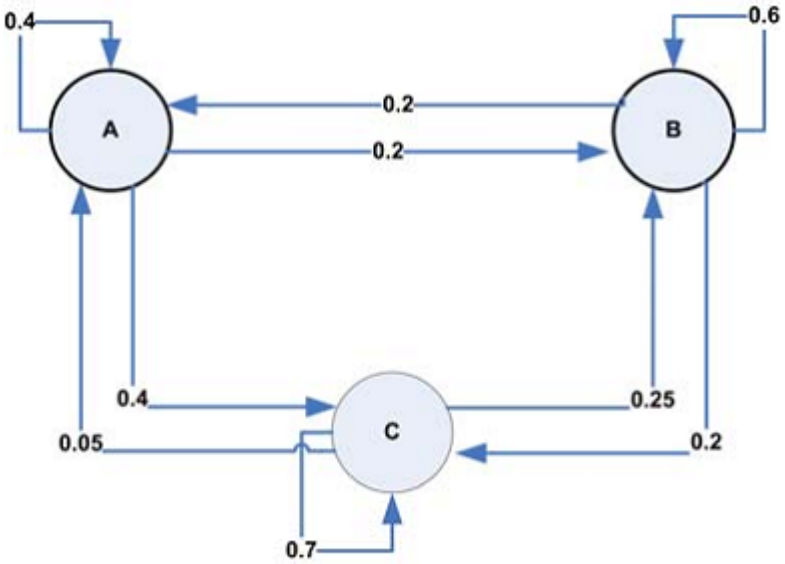


Figure 1. Hidden Markov Model

Modeling Of HMM: In order to model a process with an HMM, the following elements should be available

1. The number of states in the model, N.
2. The number of observation symbols, M.
3. The state transition probabilities described by a square matrix, A, such that,

$$A= [a_{i,j}]$$

$$a_{i,j} =p(q_{t+1}=_j \mid q_{t=i}) \ ,1<i, j<N$$

where q_t denotes the current state of the HMM. The transition probabilities of the matrix A should satisfy the following stochastic constraint.

$$a_{i,j\geq 0,} \qquad 1<i, j<N \quad \text{and}$$

$$\sum a_{i,j=1,} \qquad 1\leq i,j \leq N$$

A probability distribution in each of the states describing the occurrence of observable evidence as given by $B=\{b_j (k)\}$ refers to the probability associated with the observation of symbol k in state j, and is given by

$$b_j(k)= p(o_{t}=v_{k}| q_{t}=j) \quad , \quad 1\leq j \leq N \quad , \quad 1\leq k \leq M$$

The Matrix B should satisfy the following stochastic

$$b_j(k)\geq 0 \ , \qquad 1\leq j \leq N \quad , \quad 1\leq k \leq M$$

And $\sum b_j(k)=1 \qquad 1\leq j \leq N$

A probability distribution of the model's initial state, denoted by π_i, which is a row stochastic matrix. The three row matrices A, B and π_i define the Hidden Markov Model:

$$\lambda= (A,B,\pi)$$

In addition to the Markov condition, for mathematical and computational tractability, the following assumptions are held:

(1) State transition probabilities are independent of the actual time at which a transition takes place. This is given by:

$p(q_{t1+1}=_j \mid q_{t1=i}) = p(q_{t2+1}=_j \mid q_{t2=i})$, $1<i<N$

(2) Observation symbols are statistically independent of previous observations

$O= (o_1,o_2,o_3,\ldots\ldots,o_n)$ Thus for a sequence of observations we have

$P\{O\mid q_1,q_2,q_3\ldots.,q_n , \lambda \} = \pi P(O_{n\mid} q_{n,} \lambda)$.

Operations:

Operations with Hidden Markov Models are often carried out with three goals in mind: (1) given the model , $\lambda=(A,B, \pi)$ and a sequence of observations O, one might want to compute the likelihood of the observed sequence O,

(2) given $\lambda=(A,B, \pi)$ and an observation sequence O, one might want to determine an optimal state sequence for the underlying Markov process – in other words, uncover the most likely hidden states of the HMM;

(3) given an observation sequence O and the dimensions N and M $\lambda=(A,B, \pi)$ that maximizes the probability of O – this is useful for one might want to determine training the model to best fit the observed data.

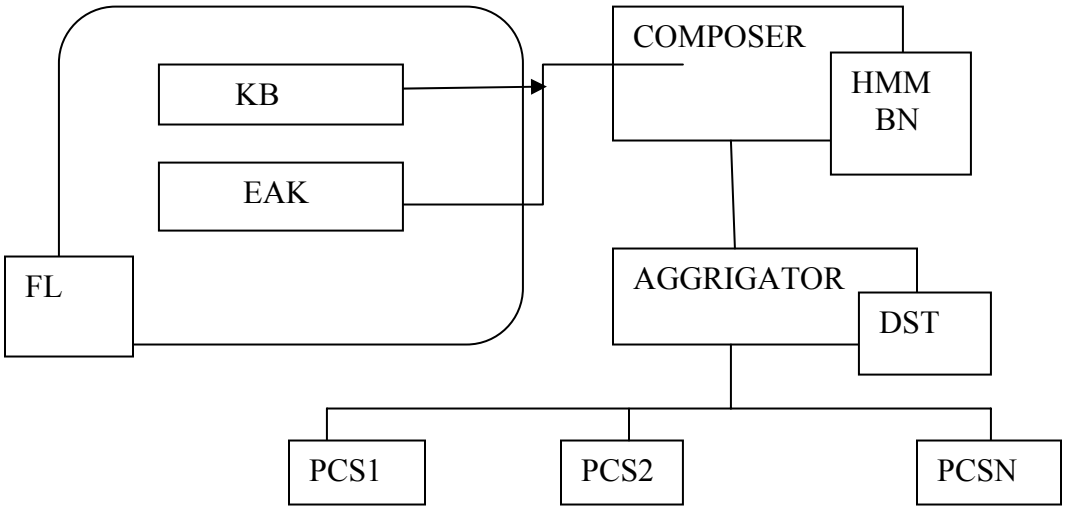


Figure 2: Architectural implementation summary

Hidden Markov Models and Bayesian Networks, both of which employ Baye's theorem and the Markov condition to model conditional dependencies among random variables. Both schemes, besides being well-established recognition schemes in various electrical engineering and computer science fields, have several efficient algorithms that yield approximate, tractable results. A Hidden Markov Model represents stochastic sequences as Markov chain, the states of which are associated with observable evidences. Desirable features include: given the model and a sequence of observations, it is possible to compute the likelihood of the observed sequence; and, given the model and an observation sequence, it is possible to compute an optimal state sequence for the underlying Markov process, that is, it is possible to uncover the most likely hidden sequences of the model. These two characteristics make HMM suitable for implementing a Composer, which takes primitive contexts as observable sequences to 'uncover' a real-world situation. The problem with HMM is that, more often than not, human activities as well as most real-world situations are quite difficult to describe as a sequence of events. Besides, even if we manage to model human activities as sequence of events, it is difficult to grant mechanisms for capturing every element of the observable sequence; hence, it is costly to predict the behaviour of the model when one or more of the observable sequences are missing. Bayesian Networks exhibit many features which make them suitable for implementing a Composer. To begin with, since the model can encode dependencies among primitive contexts, it can readily reason about situations where some data entries are missing; secondly, it is an ideal representation for combining prior knowledge and sensory data; and thirdly, the network can be trained to learn causal relationships between primitive contexts and the real-world situation they describe. Figure 4.5 shows the components of the context processing architecture along with the corresponding estimation and recognition schemes. In the next chapter, we will illustrate how we employed Bayesian Networks to implement a software Composer.

Authoring tools for HMM:

1. Hidden Markov Model toolbox

2. MAT lab

CHAPTER 3

Architecture of Synapse: The Synapse consists of four parts: 1) the sensor event collection part that captures real world information, 2) the service control part that provides services, 3) the Synapse Core, and 4) the user interface. Architecture of Synapse is shown in Figure 3.

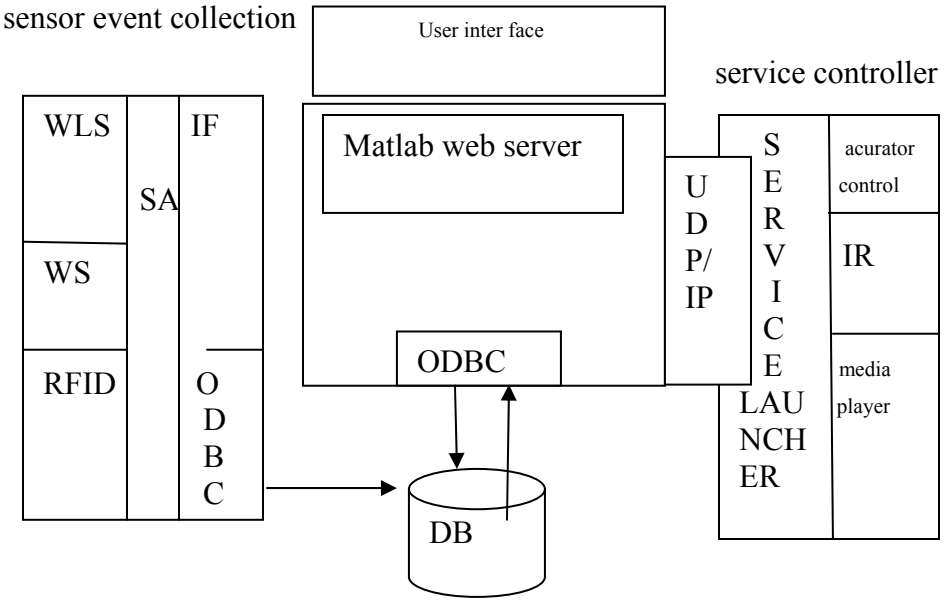


Figure 3.Architecture Of Synapse

WLS: Wire Less Sensors; WS: Wired Sensors

SA: Service Aggrigator; IF: Interface; IR:IR Remote control

Sensor Event Collection Part: The sensor event collection part captures real world information from various sensors, converts raw data into useful contexts (we call them "sensor events" in Synapse), and records these sensor events in database. Sensor Aggregator fuses the raw sensor data and reduces the noise. For instance, the average temperature in a room is fused from different temperature sensors. Context Inference

extracts complex events such as “the user is sleeping” from simple events. On this test bed, 4 kinds of sensors are used 3to produce 11 events: RFID is used to identify users U^3 wireless sensor nodes [9] are used to capture the temperature, brightness and human motion, a contact detector detects whether the phone is in use, and an e-calendar detects a day of the week. All the sensor events are recorded as a time series $\{E_1, E_2\ldots\}$ in database. Each sensor event is recorded as E (EN, EV, ET), which respectively represents the event ID, the event value and the time at which this event is recorded. We, predefine a set of events $\{e_{1,}. \ldots e_N\}$ (such as e1 means “temperature”, e2 means “brightness”),and EN €$\{ e_1,..,eN\}$. Many context inference schemes can be used to recognize events and their values from raw sensor data [9]. However, since event values are generated from different types of sensors (e.g. the temperature is 25C, and the humidity is 60%), and it is difficult for a general core to process all types of values, we use fuzzy sets [8] approaches to unify all the event values between 0 and 1 as in [9], which means EV € [0, 1]. Basically, an event will be recorded when the value changes. However, in many scenarios, it is not necessary to record events as frequently as they change, so we can add some requirements to event recording. Events will not be recorded, until they satisfy these requirements. (e.g. one requirement is “e_1 is over 0.7”, so e1 will not be recorded until it is over 0.7.)

Service Control Part: The service control part controls various devices to supply services. Service Launcher operates as a proxy between Synapse Core and the devices. It can receive a service ID from Synapse Core through UDP/IP networks, and controls the device corresponding to this service ID. It can also send the ID of a selected service to Synapse Core for service recording. As a result, it is easy for Synapse to add new services, since Synapse Core can manage them with only IDs, and ignore the various operations of different devices. On this test bed, 4 devices are used to provide 23 services: a light and a fan provide on/off services, a TV provides on/off, 12 channels and 2 videos, and a music player provides on/off and music mute/loud services. All the services are recorded as a time series $\{S_1, S_2,\ldots\}$ in database. Each service is recorded as S (SN, ST), which respectively means the service ID, and the time at which, this

service is recorded. We predefine a set of services {s_1,s_2, ,… ,s_M } (such as s1 means "turn on light", s2 means "mute music") and SN €{ s_1,s_2, ,… ,s_M}.

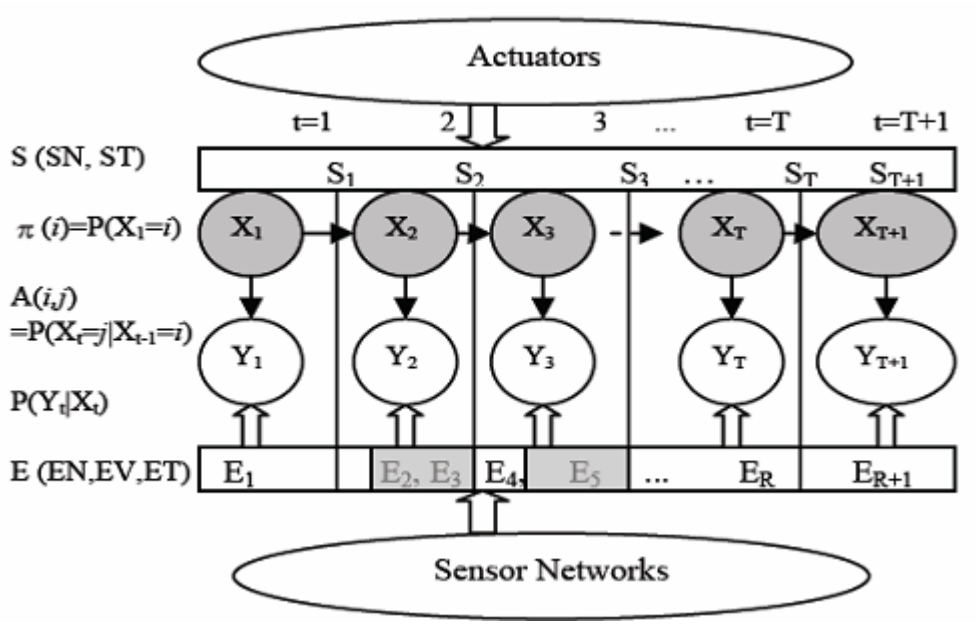
Synapse Core: We apply HMM to model the relationship between the sensor events and the services. Figure 2 shows one cycle of Synapse model. There are two basic components in HMM and the hidden state Xt observation of state Yt. In Synapse, each hidden state Xt corresponds to a service St(not lower case s), to indicate the situation in which this service is used, and the observation Yt is a vector of even values (y_1,y_2,…,y_N), which are the current values of {e_1,e_2,,…,e $_N$}. HMM is a five-tuple (Omega _X, Omega_ O, A, B,. Let lambda = {A, B, pi} denote the parameters for a given HMM with fixed Omega _X and Omega _O.

Notational conventions:

T = length of the sequence of observations (training set)
N = number of states (we either know or guess this number)
M = number of possible observations (from the training set)
Omega _X = {q_1, q_2, q_3….,q $_n$ } (finite set of possible states)
Omega_O = { o_1,o_2,o_3,……,o_n} (finite set of possible observations)
X_t random variable denoting the state at time t (state variable)
O_t random variable denoting the observation at time t (output variable)
sigma = o_1,o_2,o_3,……,o_T (sequence of actual observations)

Distributional parameters:

A = {a_{ij}} such that a_{ij} = Pr(X_{t+1} = q_j |Xt = q_i) (transition probabilities)
B = {b_i} such that $b_i(k)$ = P(O_t = v_k | X_t = q $_t$) (observation probabilities)
π = Pr(X_0 = q_i) (initial state distribution)



.

Figure 4: One Cycle Of Synapse Model

In the Executing Phase (t>T), Synapse uses the learned transition matrix $A_{(i,j)}$=P(Xj|X i), observation model P(Yt|Xt) and the current observation Yt to compute the occurrence probability of each service. A two-step filtering algorithm is applied: in update step, the probabilities of current state can be gained as we compute P(Xt |Yt) in predict step, the probabilities of next state can be predicted as we compute $P(X_{t+1} |Y_t)$. As a result, the occurrence probability of each service can be computed as the occurrence probability of each state corresponding to these services. After that, we can sort the services in a descending order of probability. If a probability is higher than a user-defined threshold, the corresponding service will automatically start in Active Mode. The top 5 services will be recommended as a list to the user interface in Passive Mode. Passive Mode is mainly used in Synapse. All these algorithms are implemented on MATLAB.

User Interface : Synapse provides a user interface in XML form on MATLAB Web Server. Users can browse this web through PC, PDA, or cellular phone, and start a service by selecting the service ID. The recommended service list on this web can automatically update after a fixed interval, or be manually updated by users.

Preliminary Evaluation Of Synapse : In order to examine the practicability of our methods, we implemented three simple scenarios on the smart car test bed, and preliminarily evaluated Synapse on three aspects:1) feasibility of Synapse, which means whether Synapse can successfully provide services based on the learned habits and the current sensor events, 2) time complexity of algorithms, which examines whether it is practically quick enough to gain the results, 3) correctness of the recommendation, which examines whether the results of prediction are practically accurate enough. Feasibility of Synapse Using the sensors and devices, we collected 100 training samples: 25 of which are “seat vibrating” scenario using “seat vibrating On(10) ” and “seat vibrating off(15)" services, 15 of which are warning messages on(5) and off(10) 25 of which are “left indicator” scenario using “ left indicator on(13)” and “left indicator of(12)” , and 35 of which are “ side parking” scenario using “side parking activate(23)” and “side parking deactivate(12)” services. Each sample is a combination of one service and a group of sensor events. For instance, in “seat vibration” scenario, when a user was in the sleeping state or drowsing state, he selected “seat vibration on” service, so the user’s ID, the condition of user and the “seat vibration on” service were recorded as one training sample. We used such training samples to learn users’ habits in three scenarios. After learning, we changed the status of users and environment, and Synapse successfully provided dynamic services adapting to the changed situation. For instance, in “side parking ” scenario: when we was using the phone, Synapse provided “side parking indicator on” to turn parking the car, when we finished using the phone, Synapse provided “side parking off” to pick up towards journey. These were collected as test samples, which were used to examine the correctness of recommendation.

Implementation Using MATLAB with HMM TOOL KIT

TRANS=

[.50,0.40,0.05,0.05;0.05,0.40,0.40,0.15;0.05,0.05,0.60,0.30;0.05,0.05,0.30,0.60;];

EMIS = [1/2,1/2;1/2,1/2;1/2,1/2;1/2,1/2];

[seq,, states] = hmmgenerate(100, TRANS, EMIS);

a) Transition Matrix

	1	2	3	4
1	0.5	0.4	0.05	0.05
2	0.05	0.4	0.4	0.15
3	0.05	0.05	0.6	0.3
4	0.05	0.05	0.3	0.6

b) Emission Matrix

	1	2
1	0.5	0.5
2	0.5	0.5
3	0.5	0.5
4	0.5	0.5

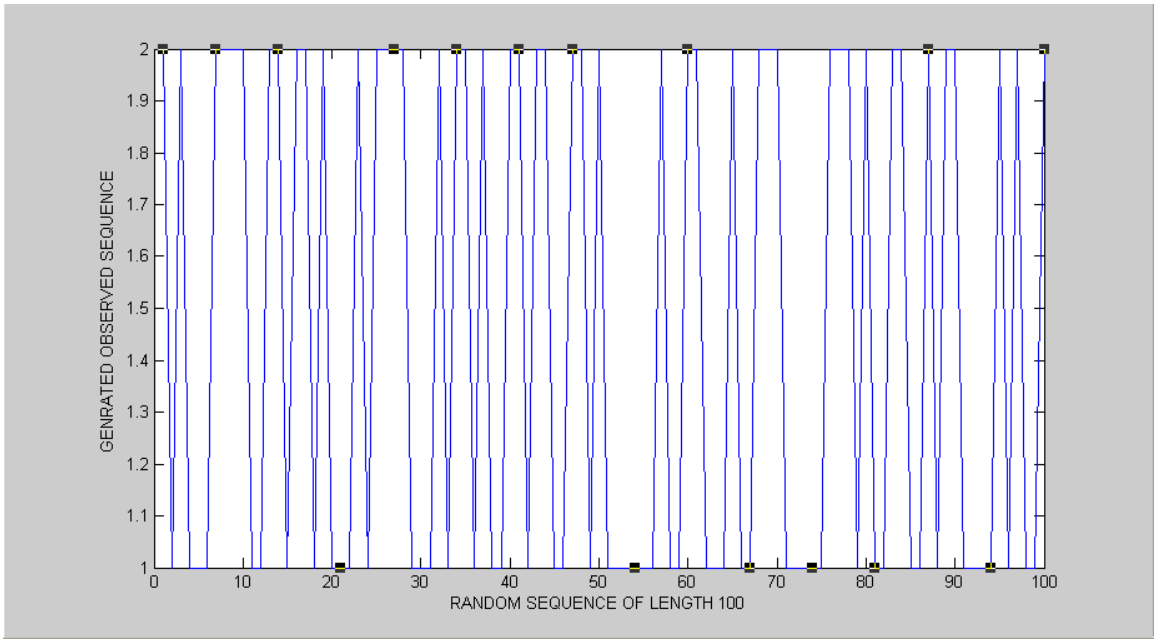


Fig 5: Genrating observed sequence for given random sequence

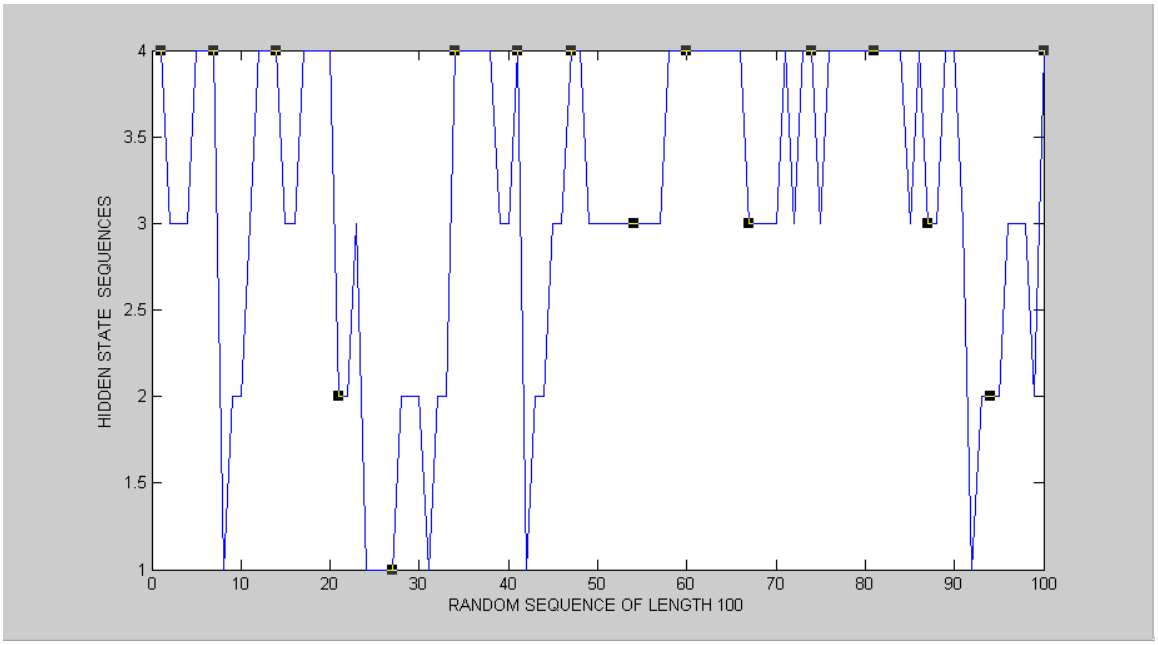


Fig 5: Genrating hidden state sequence for the random sequence

RESULT ANALYAIS: In the above graphs we can observe that many observed sequences fall in the 4th state known as "side parking", this is because as we had taken more samples from the side parking scenario. So we can analyze that the synapse will provide the service depending upon the service which he can use frequently. So by employing suitable transition matrix to our required system we can easily estimate the hidden states and with these states we can estimate and can make the decision weather our synapse core working properly or not to the respective situation.

CONCLUSION AND FUTURE WORK: In this paper, we presented a context-aware service platform – Synapse. By exploiting the recorded histories of contexts and services, Synapse can learn the users' habits. After that, Synapse can predict the most relevant services that users will use in current situation based on their habits, and provide services in Active Mode and Passive Mode. We described our algorithms and the implementation of synapse in detail. The preliminary evaluation with real world data revealed that Synapse was practicable and should be built at car. Now we are extending the sensor and service parts and implementing an entire Synapse system in a car. The experiment with real inhabitants will be conducted in the future.

In future work we can employ these scenarios in the smart car system. This is achieved By using the Fuzzy Logic,HMM,and with suitable sensors.

REFERENCES:

1. Aware Home. http://www.cc.gatech.edu/fce/ahri/.

2. Barkhuus, L. Is Context-Aware Computing TakingControl Away from the User? Proceedings of Ubicomp2003, LNCS 2864.

3. Hamilton, J. Time Series Analysis. Wiley, 1994.

4. Korpipä, P. Bayesian approach to sensor-based context awareness. Personal and Ubiquitous Computing, Vol. 7Issue 2, July 2003.

5. Rabiner, L. R. A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings of the IEEE 1989, 77(2):257–286.

[6] Dey, A.K., and Abowd, G., "Towards a Better Understanding of Context and context-Awareness", CHI Workshop, 2000.

[7] Rabiner, L. R. A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings of the IEEE 1989, 77(2):257–286.

[8]. Richard Etter, Patricia Dockhorn Costa, Tom Broens "Context-Aware Notification Services"

[9]. Barkhuus, L. Is Context-Aware Computing Taking Control Away from the User? Proceedings of Ubicomp 2003, LNCS 2864.

[10]. Das, S. The Rule Based Approach Towards Context-Aware User Notification Services IEEE 1-4244-0237-9/06 Dec. 2006.

.