A Context Inference Framework based on Fuzzy Colored Timed Petri Nets

KEON MYUNG LEE[1], KYOUNG-SOON HWANG[1], CHAN HEE LEE[2]
[1]School of Electrical and Computer Engineering, [2]Dept. of Microbiology,
Chungbuk National University, Korea

Abstract[1] - *In context-aware computing environment, some context is characterized by a single event, but many other contexts are determined by a sequence of events which occur with some timing constraints. Therefore, context inference should be conducted by monitoring the sequence of event occurrence along with checking their conformance with timing constraints. Some context could be described with fuzzy which can be easily used in situation description. Multiple entities may interact with a service system, and thus the context inference mechanism should be facilitated to handle multiple entities in the same situation. This paper proposes a context inference model which is based on Petri net. The model represents and handles the sequential occurrence of some events along with involved timing constraints, deals with the multiple entities using the colored tokens, and employs the concept of fuzzy tokens to support the fuzzy concepts.*

1 Introduction

With the advances in information technologies, sensor technologies, various pilot context-aware service systems have been developed for context-aware information services in ubiquitous computing environment, intelligent services in intelligent robots, and various other intelligent application software.[1-9,11,16] In order to provide quality services, it is valuable to make use of contextual information. Hence, the quality service systems need to collect sensor data from which context could be deduced and to have a mechanism to conduct such context inference. Context denotes any kind of information which can be used to characterize the state of an entity. An entity can be either user or anything involved in the interactions between user and the application.[1] Typical examples of contextual information include location, identity, time, behaviors and so on which give some clues to answer the questions about *who*, *what*, *when*, and *where*. An application which takes into account the working context in providing information or services is called *a context-aware system*. The data used in determining some context are collected by sensors and transmitted to a context-aware service system through sensor networks. Some data are inherently stream data since the sensors continuously monitor some interesting aspects of the environment. On the other hand, each sensed datum can be regarded as an event in the environment. Some context can be determined by a specific single event, yet in many situations a collection of events satisfying some sequential and timing constraints determines a context. Therefore, context inference is involved with the task to effectively detect a specific occurrence sequence of events and the task to check whether those events satisfy the specified timing constraints. To our best knowledge, the existing context-aware service systems have not explicitly taken in consideration such kind of context inference.

This paper is concerned with a Petri net-based context inference model called fuzzy colored timed Petri net model which allows to easily represent complicated contexts which are determined by a sequence of events with timing constraints, enables to model multiple objects located in the same situations, and in addition incorporates the notion of fuzzy tokens to deal with fuzzy context.

The remainder of this paper is organized as follows: Section 2 presents some related works including Petri net models and existing context inference approaches. Section 3 introduces the proposed fuzzy colored timed Petri net model. Section 4 shows how to model the contexts, and in final it addresses the conclusions in Section 5.

2 Related Works

2.1. Petri Net Models

Petri nets are one of well-known tools used for modeling, formal analysis and design of systems. A Petri net is a bipartite directed graph defined as a tuple $<P, T, F, M>$, in which $P = \{P_1, P_2, \dots, P_n\}$ is a finite set of places with $n \geq 0$; $T = \{T_1, T_2, \dots, T_m\}$ is a finite set of transitions with $m \geq 0$ and $P \cap T = \emptyset$; $F \subseteq (P \times T) \cup (T \times P)$ is the flow

This work was supported by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD) (The Regional Research Universities Program/Chungbuk BIT Research-Oriented University Consortium)

relation, a mapping representing arcs between places and transitions. M: $P \rightarrow I$, $I = \{0, 1, 2, \ldots\}$, is a function that associates a number of tokens to places.

A Petri net is graphically visualized by representing its places with circles, transitions with bars, arcs with arrows, and tokens with small black dots. A place containing one or more tokens is said to be marked. When each place incident on a transition is marked that transition is said to be enabled. An enabled transition may fire by removing one token from each of its input places and putting one token into each of its output places. In the basic Petri net tokens have no identity. From the basic Petri net, several extended Petri nets have been proposed, which includes colored Petri net, timed Petri net, colored timed Petri net, object-oriented Petri net, and so on.[10,13,15]

A colored timed Petri net(CPN) is defined as a tuple $< \Sigma, P, T, F, \tau >$ in which Σ is a finite set of token colors; P, T and F are the same with those of the classical Petri net, respectively. τ: $T \rightarrow \{0, 1, 2, \ldots\} \times \{0, 1, 2, \ldots, \infty\}$ is a function mapping from each transition to a pair of values corresponding to *release time* (i.e., delay) and *maximum latency* (i.e., timeout), respectively. For any transition $t \in T$, we write $\tau(t) = (\tau^r, \tau^m)$ and we require that $\tau^r \leq \tau^m$.

In the colored timed Petri nets, tokens have color types and a token of one color is discernible from a token of another color. Within a color class, an individual token cannot be distinguished from one another.

2.2 Context Inference Systems

There have been various efforts to support context inference for context-aware service applications. The Stick-e Notes system[2] is a generic framework to help develop context-aware applications, on which they represent the context-aware knowledge using *if-then* rules and apply the inference mechanisms (e.g., forward or backward chaining) to conduct context inference based on the rules. Schilit's System Architecture[3] is an architecture to support context-aware mobile computing applications which is focused on the contexts of users and devices. The architecture contains the following interacting components: device agents to manage the states and functions of devices, user agents to take care of users' preference, and the active map to keep track of the location information for the users and the devices. The architecture allows the restricted forms of context inference because it is designed to care about the context between users and devices. CALAIS[4] is an architecture for the context-aware applications which are mainly focused on location-related contexts. It is not so flexible to handle various kinds of contexts. TEA project[5] is an architecture designed to support context-aware services on personal mobile devices. It employs a blackboard model for context inference in which sensors record their acquired data into the blackboard, context interpreters get the summarized information about their concerned data or interpret them and then they also store the processed information into the blackboard, the application implements context-aware services by referring to the contextual information from the blackboard. Due to restricted flexibility of the blackboard model, the TEA project architecture has difficulty in handling the type of context inference that we have interested in.

3 The Proposed Fuzzy Colored Time Petri Net Model

3.1 The Context Inference

In the applications to have to catch the context from the data acquired by the sensors, some context is determined by a specific event, and other types of context are characterized by a collection of events that happen in a way to satisfy the specified timing constraints among them. For example, suppose that a person had turned the gas range on and went out home, where we consider the situation as being in danger of fire if a specified amount of time has passed yet the person did not come back. In order to provide contextual services, it is needed to have an effective way to describe contexts of interest, and to have an efficient inference mechanism to determine the valid contexts from available data some of which are stream data. As an effective framework for the context representation and the context inference, this paper introduces a new model named *fuzzy colored timed Petri net*.

3.2 Context Representation based on the Fuzzy Colored Timed

For the convenience of description, let us denote as a single event-based context a context which is characterized by a single event, and as a multiple event-based context a context which is characterized by a collection of events which satisfies some timing constraints.

The *if-then* rules which are widely used in knowledge representation could be employed to represent some sort of contexts. The *if-then* rules allow to easily describe some simple contexts, but have difficulty in expressing complex contexts like multiple event-based contexts. Even though a multiple event-based context is expressed in *if-then* rules, it is hard to follow the progress of the contexts in the *if-them* rules on the fly.

Procedural modules can be employed to recognize multiple event-based contexts. This approach has flexibility in handling context inference, yet it also implies complexities because we have to build procedural modules for every single context and to pay special attention to possible conflicts among those modules. In addition, since events should be delivered to each procedural module, it could cause the structure of the implementation to be entangled. On the other hand, it is not easy to keep track of context progress with which we are in part concerned in this study.

In order to handle context inference, the proposed method adopts a colored timed Petri net-based model to describe contexts and to conduct inference. In a Petri net model to represent a context, a place may correspond to an event and transitions are used to describe the timing relationships among events. When a specific event happens, the corresponding place turns out to have a token. Each transition t is bound with a timing constraint label $[\alpha^t, \beta^t]$, where α^t is the release time, and β^t is the latency time. The release time α^t indicates that the transition is enabled after α^t time under the condition that all its incoming places have their token. The latency time β^t indicates the transition t to be fired after β^t time under the condition that t has been enabled. It means that a transition t is enabled when all its incoming places have had token for α^t time and then is fired after β^t time.

There may be multiple entities to participate in the environment. Therefore, the colored tokens are used to discriminate each entity that plays in the same context. When it is needed to discriminate entities, it is designed to put colored tokens into the corresponding places at its event occurrence and to make the corresponding transitions produce the corresponding tokens. The sink places containing a circle indicate the place in which tokens arrived are just discarded.

In the proposed context inference framework, we classify the events into triggering events and state-changing events.

Def. Triggering events and State-changing event

The *state-changing events* denotes ones which change the state of an entity or the environment and the changed state keeps persistent until its state is reset, e.g., *the light is on*. The triggering events are ones which do not make any persistent state change, e.g., *the bell rings*.

As a matter of fact, a state-changing event consists of a triggering event and a reset event and thus can be modeling with triggering events. For the convention of context modeling, the proposed framework uses the notion of state-changing events.

Def. A context CT with 2 events is defined by $CT = ((e_1, e_2), C(e_1, e_2))$, where $e_i \in TE \cup SE$, TE is the set of triggering events, SE is the set of state-changing events, and $C(e_1, e_2) \in \boldsymbol{C}$ is the timing constraints on the events e_1 and e_2 which are expressed in the sequel $\boldsymbol{C} = C_1 \cup C_2 \cup C_3 \cup C_4$.

Due to two kinds of events, such contexts with two events can be categorized into *TE-TE* contexts, *SE-SE* contexts, *TE-SE* contexts, and *SE-TE* contexts. For *TE-TE* contexts, there are two related triggering events e_1 and e_2, their timing constraint is expressed as follows:

$$C_1 : TE \text{ x } TE \to (T_{e1}, T_{a1}, T_{e2})$$

In the above, T_{e1} is called the *exclusive time* of event e_1 during which the following event e_2 should not occur, and T_{a1} is called the allowance time during which e_2 should happen. In a similar way, the other constraints are described as follows:

$$C_2 : SE \text{ x } SE \to (T_{se1}, T_{sv1}, T_{sa1}, T_{se2}, T_{sv2})$$

For a state-changing event, T_{se1} is the exclusive time in which the changed state is kept and the following event is not allowed to occur, T_{sv1} is the valid time interval $[T_{svl1}, T_{svu1}]$ in which the state needs to be reset, T_{sa1} is the allowance interval $[T_{sal1}, T_{sau1}]$ for the following event to have to occur, T_{se2} is the exclusive time of e_2 and T_{sv2} is the valid time of e_2. When the reset event is not needed to describe context, the valid time interval constraint T_{svi} may be omitted.

$$C_3 : TE \text{ x } SE \to (T_{e1}, T_{a1}, T_{se2}, T_{sv2})$$

$$C_4 : SE \text{ x } TE \to (T_{se1}, T_{sv1}, T_{a1}, T_{e2})$$

The exclusive time T_{sei} is measured from the occurrence of the event, the valid time T_{svi} for a state-changing event is measured from the end of the exclusive time, and the allowance interval $T_{sai} = [T_{sali}, T_{saui}]$ is measured from the end of the exclusive time.

The contexts made of two events can be modeled by the Petri net models as follows: Figures 1 to 4 show typical Petri net models for *TE* x *TE* contexts, *SE* x *SE* contexts, *TE* x *SE* contexts, and *SE* x *TE* contexts, respectively

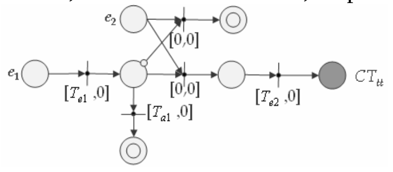

Figure 1.A *TE-TE* context model $CT_{tt}(((e_1, e_2), (T_{e1}, T_{a1}, T_{e2}))$

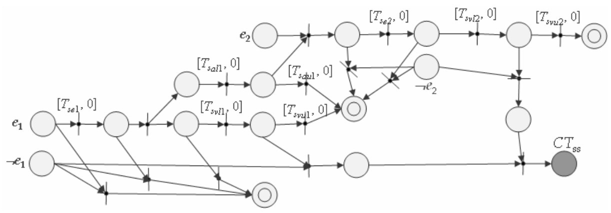

Figure 2. An *SE-SE* context modelC$_{ss}$$(((e_1, e_2), (T_{se1}, T_{sv1}, T_{sa1}, T_{se2}, T_{sv2}))$.

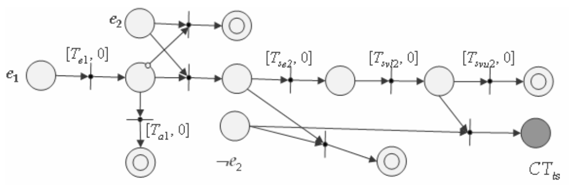
Figure 3. An *TE-SE* context model $CT_{ts}(((e_1, e_2), (T_{e1}, T_{v1}, T_{se2}, T_{sv2}))$.

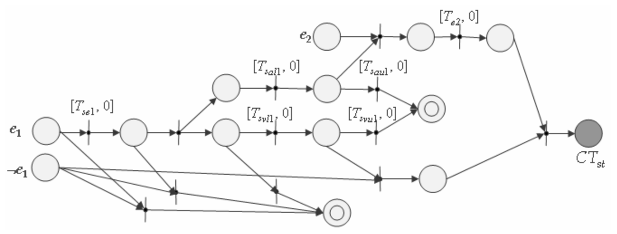
Figure 4. An *SE-TE* context model $CT_{st}(((e_1, e_2), (T_{se1}, T_{sv1}, T_{sa1}, T_{e2}))$.

As we have seen above, the proposed Petri net-based representation allows us to express various contextual situations in which multiple events are involved and on which some timing and sequential constraints are imposed. Single event-based context can be modeled by a single place to which context-aware service module is bound and thus when the corresponding event happens, its service is made to be triggered. By composing the above-mentioned primitive context modules in their own way, the developers can easily represent the complicated contextual situations. The visual representation of Petri net helps developers easily verify and understand what context is designed and how it works. In addition, it allows to visually keep track of the context progress responding to the incoming stream of sensed data.

3.3 Fuzzy Context Representation

In the proposed model, there are two types of places according to how the state of places is maintained: *event state places* to represent whether a specific event occurs, and *persistent state place* to continuously provide the probably changing values of the corresponding state. In the case of an event state place, when its event happens, a token is placed into the corresponding place and once one of outgoing transitions is fired, the token is discarded. On the contrary, the tokens of the persistent state places are not discarded even though some of their outgoing transitions are fired. The tokens of persistent state places have some numeric values reflecting the state of a specific stream data from a sensor.

The proposed model uses a persistent state place to represent a fuzzy constraint. For example, when an event state '*temperature is low*' is modeled, it is represented by a persistent event place to which a stream of temperature values acquired from a thermal sensor is bound and into which a membership function expressing the fuzzy constraint *low* is embedded. The persistent event place keeps providing the membership degree of the temperature values received to the fuzzy constraint *low*.

Figure 5 shows an example of a persistent state place for a fuzzy constraint '*temperature is low*'. The place has the assigned membership function μ_{low} of fuzzy constraint low and keeps changing the value of its token according to the membership degree of the just received temperature values to the membership function μ_{low}. On the other hand, as shown in Figure 5 (a), the immediately following transition of a persistent state place is labeled with a threshold value θ and the value pair $[\alpha, \beta]$ as its timing constraint. The transition turns into the enabled state if its incoming place has been the token value greater than or equal to θ for more than α time, and is triggered when β^t time has passed after its enablement. Each time the membership degree falls below the threshold, the timer for monitoring place enablement is re-started as shown in Figure 5 (d). When the transition is triggered, it produces a token into its following place(s). According to the developer's design decision, the token may have selectively either the average membership degree over the enabled period or the final membership degree, or some other value. The token value, if exists, of the following places could be either used or discarded depending on the designed model specification.

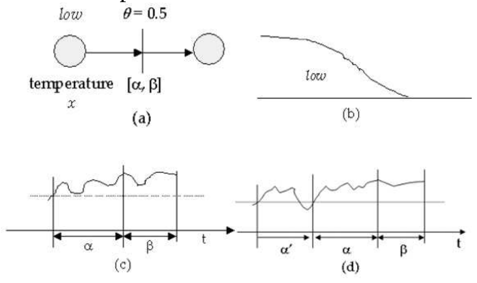

Figure 5. A fuzzy persistent state place and its corresponding transition

Once a transition with an incoming persistent state place is triggered, it starts over its mission by resetting its timers and monitoring the token value of the incoming place. According to the design decision, such a transition is optionally set to be in the sleep mode for a specified time period after its triggering.

3.4 Composition of Context Inference Models

In Section 3.2 we have described how to model contexts with a sequence of two events with timing constraints. Those 4 types of context models play the role of basic building blocks in the proposed context inference framework. When a context could be characterized with more than two events, each pair of events related with timing constraints is modeled with a basic building block. Some context could be described with multiple sub-contexts which need to be satisfied simultaneously or selectively. For the simultaneously satisfying sub-context components, the Petri net models for them are combined by directing their final places into a transition as shown in Figure 6. When each of multiple context components describes a context, their final places are merged into a place as shown in Figure 7.

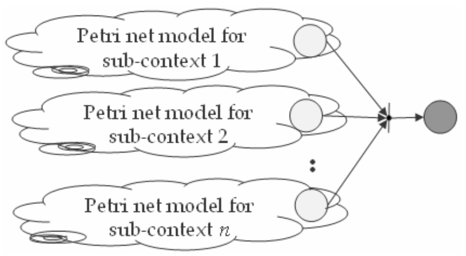
Petri net model for sub-context 1
Petri net model for sub-context 2
Petri net model for sub-context *n*

Figure 6. Composition of the sub-context components that need to be satisfied simultaneously

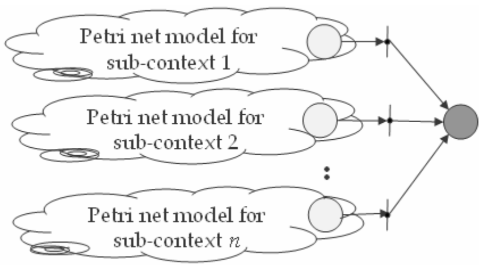
Petri net model for sub-context 1
Petri net model for sub-context 2
Petri net model for sub-context *n*

Figure 7. Composition of the sub-context components each of which can characterize the designed context

An event might be used in multiple context modules and thus the place bound to an incoming event e_i is connected to all the places for e_i in the built context modules as shown in Figure 8.

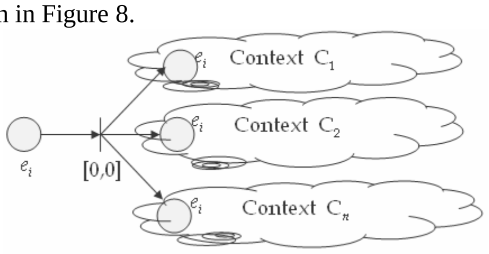

Figure 8. Binding an incoming event place with the corresponding event places of the models

Figure 9 shows an example of context modeling which describes the contextual situation that "*It is dangerous to leave the house for more than 30 minutes after one has turned the oven.*"

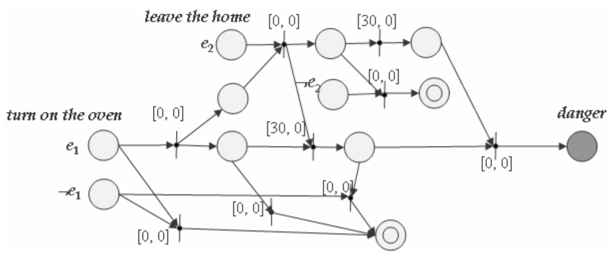

Figure 9. A Petri net model for a *danger* context

3.5 Context Inference Module

In the proposed context inference framework, each transition has an assigned processing module with which context inference is conducted. The processing module of a transition t starts the timer for α^t once all its incoming places have a token with the same color, if tokens are colored. If the transition has a persistent event state, it starts its timer when the token value becomes greater than or equal to the labeled threshold. In the middle of timer working, if some of its incoming places loses its token or the token value of its incoming persistent event place falls below the threshold, the timer is reset and stopped. When the timer reaches at time α^t, the transition turns to be in the enabled state and the timer for β^t starts. After that, when β^t time has passed, the transition is triggered and the tokens of the incoming place(s) is removed, if they are not a persistent event place, and tokens are added into the succeeding places. Along with that, the timers are reset and the processing module of the transition starts over its mission. When a token is added into a place corresponding to a context, it is regard as that the context is recognized and the bound service module is invoked for context-aware service.

4 Development of Context-aware Service Applications

When a context-aware service application is developed based on the proposed method, it is modeled in the following sequence:

Step 1. Enumerate the kinds and characteristics of available sensor data.

Step 2. Identify all contextual situations under considerations.

Step 3. Model each contextual situation with a fuzzy colored timed Petri model according to the method presented in Section 3.

Step 4. Implement the service modules for each contextual situation and bind them to the corresponding context places.

Step 5. Combine the places corresponding to the same event into a single place, if possible.

For each transition *t*, install as many pairs of timers as the number of token colors used, where timers correspond to release time α^t and latency time β^t, respectively. In order to enable flexible context-aware services, it allows transitions to use software modules which may takes as input the external sensor data and/or the output of other modules.

In order to see its applicability, we implemented a prototype system for the proposed method. In the prototype implementation, the modules bound to the places to get inputs from external sources are implemented with a multi-thread architecture, and the token management for the other places is sequentially conducted according to the topology of the designed Petri net model. Each transition is organized to have an own housekeeping module which takes care of both its timers and the processing module invocation according to fan-in/fan-out topology. These housekeeping modules are implemented to be processed in a round-robin fashion. To help track of the progress of concerned contexts, a simple visualization module to show how the tokens are propagated was implemented. In the visualization module, it is assumed that the graphs for Petri net modules are provided by the developers. At the current stage of development, we implemented a simple Petri net drawing tool which uses an XML-based file format to store the topology of Petri net models designed by the developer. The simulation environment is compiled together with the processing modules along with an XML file formatted data which describe the models.

5 Conclusions

Various service applications to use contextual information are expected to emerge. For such applications, it is crucial to recognize the contexts from the data acquired. This paper proposed a model called *fuzzy colored timed Petri net* which allows easily to represent context recognition knowledge and to enable the inference recognition. The proposed method allows to model complicated contexts in which multiple events are involved with some timing and sequential constraints. It also allows to conduct the context inference and to visually keep track of the context progress with the inherent characteristics of Petri net models. The places and transitions corresponding to fuzzy context allow to handle flexiblely various kinds of context. Compared to the existing approaches such as rule-based systems, procedure-based systems, the proposed approach is advantageous in that it allows visual modeling and direct one-to-one mapping from models to codes. As the further studies, there remains to develop the proposed method in a software framework in a form of API(Application Program Interface) and to provide several successful application examples developed on the framework in order to show the applicability.

References

[1] A. K. Dey, “Providing Architectural Support for Building Context-Aware Applications”, *Ph.D. dissertation*, Georgia Institute of Technology, 2002.

[2] P. J. Brown, “he stick-e document: a framework for creating context-aware applications”, *Electric Publishing*, Vol.9, No.1, 1996, pp.1-14.

[3] B. N. Schilit, “System architecture for context-aware mobile Computing”, *Ph. D. dissertation*, Columbia University, New York, 1995.

[4] G. J. Nelson, “Context-aware and Location Systems”, *Ph.D. dissertation*, University of Cambridge, 1998.

[5] A. Schmidt, K. A. Aidoo, et al, “Advanced Interaction in Context”, *Proc. of HUC'99*, 1999, pp.89-101.

[6] J.-C. Na and R. Furuta, “Context-Aware Digital Documents Described in A High-Level Petri Net-based Hypermedia System”, *LNCS*, Springer-Verlag, 2004.

[7] H. Djenidi, S. Benarif, A. Ramdene-Cherif, C. Tadj, N. Levy, “Generic Multimedia Multimodal Agents Paradigms and Their Dynamic Reconfiguration at the Architectural Level”, *J. on Applied Signal Processing*, Vol.11, 2004, pp.1688-1707.

[8] H.J. Genrich, K. Lautenbach, “System Modeling with High-Level Petri Net”, *Theoretical Computer Science*, Vol.13, 1991, pp.109-136.

[9] R. Gudwin, F. Godmide, “Object Networks - A Modeling Tool”, *Fuzzy Systems Proceedings, 1998 IEEE World Congress on Computational Intelligence*, Vol.1, 1998, pp.77-82.

[10] K. Jensen, “Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use”, *EATGCS Monographs on Theoretical Computer Science*, Springer-Verlag, 1992 .

[11] J.-C. Na and R. Furuta, “Context-Aware Digital Documents Described in A High-Level Petri Net-based Hypermedia System”, *LNCS*, Springer-Verlag, 2004.

[12] H. Djenidi, S. Benarif, A. Ramdene-Cherif, C. Tadj, N. Levy, “Generic Multimedia Multimodal Agents Paradigms and Their Dynamic Reconfiguration at the Architectural Level”, *EURASIP J. on Applied Signal Processing*, Vol.11, 2004, pp.1688-1707.

[13] D. J. Carmichael, J.Kay, B. Kummerfeld, “Consistent Modeling of Users, Devices and Sensors in a Ubiquitous Computing Environment”, *User Modeling and User-Adapted Interaction*, Vol.15, No.3, 2005, pp.193-195