Formal Modeling For Pervasive Design of Human-Computer Interfaces

Ines Riahi, Faouzi Moussa
National School of Computer Sciences, Cristal Laboratory
University of Manouba, Tunisia
Emails: {ines.riahi@yahoo.fr, faouzimoussa@gmail.com}

***Abstract*—The advent of mobile interfaces induces an evolution on the Human Computer Interaction (HCI) field. We observe the emergence of several mobile devices and sensors that gave birth to the ubiquitous environments. In our research, we focus on: (i) how to adapt the interface to its environment, specifically in its context of use and (ii) what relationship has the context with the users' task. This paper will propose a formal approach for specifying user interfaces adapted to the context of use. We will focus on the strength of formal approach to context and user's modeling and how to infer users' requirements through the model of the task for critical domains. Our approach will be illustrated by a case study on the monitoring of diabetic patients.**

Keywords-pervasive user interfaces; ubiquitous computing; formal modeling; critical domains.

I. INTRODUCTION

Ubiquitous environment is a physical space in which technology is seamlessly integrated in order to assist the user in performing tasks to reach its goals more conveniently.

Ubiquitous environments are often considered highly dynamic environments and the contextual information can change at runtime. User interfaces should provide the right information for the right person at the right time [1]. In order to cope with such a complexity, new methods need to be developed. The research field of HCI has introduced many techniques for interaction design that are partially suitable for ubiquitous environment. However, it has an enormous rate of environmental information and the user's task becomes difficult to identify.

The specification of user interface adapted to the context of use presents several problems. The consideration of the user's task represents an important criterion in an environment where the context has a direct impact on the user's task. The users' interface must change according to the context and task at a specific moment.

Modeling ubiquitous environment and user's task poses several issues. In fact, pervasive environments are extremely dynamic and hold a vast amount of information. In critical domains, such as health, nuclear and transport systems, modeling pervasive system must be rigorous with minimal percentage of error risk. If the user's interface shows wrong information, it will have a disastrous impact on the user's task. So, the use of formal approach in such domains is required to guarantee a valid interface since the modeling stage. To tackle these problems, we study, in the second section, different related works in the literature of context-aware systems. In the third section, we study the state of the art of modeling context and user's task based on Petri Nets (PN). Then, we introduce, in the fourth section, our formal approach for specification of adapted user interface to the context of use. We will focus on the advantage of the formal model by representing the relation between the context's model and the user's task and illustrating how to deduce the users' requirements from the task models. This approach will be illustrated, in the fifth section by a critical case study on the monitoring of diabetic patients in a smart hospital.

II. RELATED WORK

Researchers in the context adaptation area have not introduced a generic and pragmatic definition of the notion of context. Following the study of the main definitions, we have concluded that the majority agrees on the definition proposed by Dey. For our research work, we will consider the definitions of Dey [2] and Calvary [3], which define the context as the triplet of <user, platform, environment>. These definitions help to clarify the notion of context in Human-Computer Systems (HCS).

Over the years, a large number of context-aware systems have been developed for different domains. Based on different context models, these applications are able to gather, manage, evaluate and disseminate context information [4]. Among these approaches, we mention the SOCAM (Service-Oriented Context-Aware Middleware) architecture that was especially proposed to convert the physical spaces; thus, contextual information can be converted into a semantic space and can be shared between the context-aware services [5]. Moreover, the key component in the CoBrA (Context Broker Architecture) architecture is responsible for managing and processing the contextual information while maintaining the contextual model [6]. The SECAS (Simple Environment for Context-aware Systems [7]) architecture is based on three components: context management, adaptation layer and the application core. Context management is composed by the context provider, the context interpreter, the broker and the context repository. The adaptation layer considers three type of adaptation: content, behavior and user interface adaptation. The Context Toolkit, proposed by Dey [8], provides a toolkit for the development of context-aware applications. It has a layered architecture that permits the separation of context acquisition, representation and the adaptation process. This architecture is based on: (i) Widget: a software component that provides applications with access to context information from their operating environment; (ii) Interpreter: used to interpret low-level context information and convert it into

 ISBN: 978-1-61208-353-7

higher level information; (iii) Server: a connection between the applications and the widgets.

Recent researches take into account context-awareness and complex dynamic system in which context variables change over time, such as GECAF (Generic and Extensible Context Aware Framework) [9], which uses a generic framework that supports all elements found in existing systems. Also, the conceptual architecture for Adaptive Human-Computer Interface of a PT Operation Platform (AHCI of PT platform [10]) based on context-awareness improves usability, simplifies the operation process, reduces operation complexity, provides needed information timely and properly and supports user needs diversification.

Having analyzed related work, it then becomes necessary to define comparison criteria to work out the advantages and disadvantages of each architecture. The main criteria in our research are: (i) the model of context: the use of any inappropriate model for context representation could lead to incorrect interpretation of contextual information. This could compromise the entire functioning of the architecture and provide the user with inadequate adaptations. Using a simplistic model could result in conflicts in the interpretation and the description of the current context of use. (ii) User interface validation: in critical domains, the generation of the user interface must be validate. The interface will guide the user to accomplish his task. Any errors in the interface can lead to a critical situation. Notwithstanding its obvious importance, this factor is not seriously treated in the majority of research work studied. For the first criterion, the SOCAM and CoBra architecture use ontologies for modeling context. SECAS utilize XML for context description. Furthermore, the Context Toolkit applies the Key/Value model for the specification of context. In our research, we focus on graphical modeling approach.

III. CONTEXT AND USER'S TASK MODELING BASED ON PETRI NETS

Several graphical modeling approaches to context-aware systems have been proposed, such as Unified Modeling Language (UML) [11], Object Role Modeling (ORM) [12] and Petri Nets (PN) [13]. In this section, we will focus on PN. PN, proposed by Carl Adam Petri in 1962, is a mathematically-based formalism dedicated to the modeling of parallelism and synchronization in discrete systems [13]. Recently, many context-aware systems modeling approaches based on PN have been proposed. They have been recognized as promising for the representation of context [14].

Context modeling approaches using PN differ depending on the purpose. Some authors are mainly interested in modeling the behavior of context-aware application; others try to solve the problem of time and resources in applications. There are several extensions of PN, such as: (i) Synchronized PN: Reignier introduces an approach to the representation of the context and the behavior of the application [15]. (ii) Colored PN (CPN): Silva proposes to combine 3D modeling tools with CPN for modeling 3D environments. In this model, the place is used to indicate the current state of the user and the transition is used to deduce the movement of the user and the behavior of the components [16].

The approach of modeling context must verify several requirements of pervasive environments, such as partial validation, formal language and formal verification. The last two requirements are essential in our research:

- Partial validation: It is preferable to be able to partially validate contextual information because of the complexity of contextual interrelationships, which may be responsible for any modeling error.
- Formal language: The chosen modeling method should have formal semantics. Formal methods comprise formal specification using mathematics to specify the desired properties of the pervasive system.
- Formal verification: the model must be verified through rules or mathematical property. This can be helpful in proving the accuracy of pervasive systems; this ensures the validation of user interfaces before they are implemented.

The PN-based methods have all the required characteristics. Indeed, the use of PN for modeling paves the way for formal verification and validation of the interfaces. These criteria are very important in our research work. The modeling of pervasive systems in a critical domain requires a rigorous validation of the interface in order to present the best solution to face an urgent situation.

This saves considerable time in the development cycle, particularly during the validation phase. Moreover, PN have a formal definition; they are highly capable of expressing aspects such as parallelism, timing, concurrency, etc. They possess many techniques for an automatic verification of properties. They provide, in addition, an unconstrained graphic representation.

We use "small granularity" PN ensuring the accuracy of our model and the partial validation. In fact, the context model is decomposed in small granularity and can be done through a simple validation based on partiality.

Modeling the user's task has a tough impact on the design of the user interface. In recent years, there have been different approaches to the specification of the task and how it relates to the area of application. Several notations have been proposed (ConcurTaskTrees CTT [17], Collaborative Task Modeling Language CTML [18], and PN [19]). The tasks are organized hierarchically to represent the task's decomposition, which is executed to meet a particular purpose. The process of task decomposition is ceased once the atomic task 'action' is obtained.

The PN are continuously expanding and they are a suitable tool for modeling the HCS. Initially, they were only used to describe tasks that were to be computerized. But later, especially with the emergence of High Level PN, they were used to model the HCI.

In the next section, we present our formal approach for specification of pervasive user interfaces.

 ISBN: 978-1-61208-353-7

IV. APPROACH FOR FORMAL SPECIFICATION AND GENERATION OF USER INERFACES ADAPTED TO THE CONTEXT

The overall objective of our research is to generate in real-time a user interface adapted to the current context of use in critical situations. The specifications of the HCS must consider the context modeling. As we mentioned in the third section, the captured context data will be modeled using PN. As the context is defined by the triplet <user, platform, environment>, each element will have its own PN: (i) The User's PN describes the different users of the application; (ii) The Platform's PN presents the different platforms that host our application; (iii) The Environment's PN describes the different information of the environment (i.e., geographical location, time, etc.).

Since each component of the context is modeled by its own PN, the marking of all these networks determines, at any given time, the current state of the context. Indeed, the marked places in the three networks determine the values of the triplet <user, platform, environment> and characterize the current context. Furthermore, according to the context in which the user operates, the user's task may vary. Indeed, each user task is specific to a given context. Thus, a set of pairs (context, task) will compose, among others, the model of the HCS. The user's task will also be modeled using PN. Each task will be decomposed into elementary tasks to be modeled using elementary structure of PN.

A. *Elementary structure of PN*

The modeling of an elementary structure is illustrated by Figure 1.

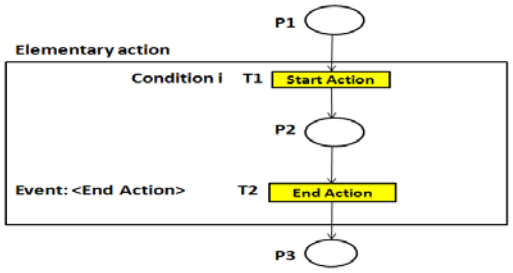

Figure 1. Structure of an elementary action

The validation of the condition i (transition T1) models the fact that the user will start the execution of the action relative to that condition. After the event, the "end action" (transition T2) expresses the fact that the user action was performed and ended. The place P2 represents a waiting state for the end of the action's execution, while the places P1 and P3 model the state of the user before and after the execution of his action. For example, P1 models the user's mental intention in order to act. The place P3 expresses his state at the end of the action's execution.

All the user's actions and components context behavior (elementary or composed) are arranged according to typical compositions: sequential, parallel, alternative, choice, iterative or of-closure. We present below the principle of parallel and alternative composition:

- The parallel composition expresses the possibility of simultaneous execution. The parallelism is ensured thanks to an input synchronization place. This place activates at the same time all the places of initialization of the parallel actions to be executed. Note that the effective parallelism can only be done if the actions to be executed do not use the same resources. Otherwise, a partial or complete sequencing would be necessary. Obviously, the number of places Pn must be equal to the number of parallel actions Ai. Thus, to ensure the parallel composition of actions, it is necessary to synchronize the places of entry and those of exit of those actions (Figure 2).

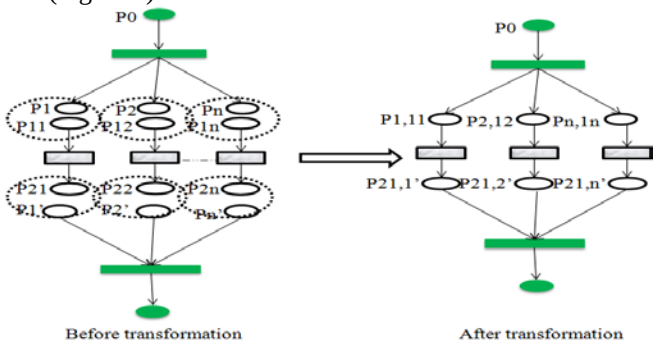

Figure 2. Parallel composition

- The alternative composition of n actions reflected a performance always exclusive of these actions. To avoid an actual conflict, conditions are associated with transitions to unambiguously determine which action should be executed. The alternative composition of n networks is realized by composing them sequentially with an ALT structure and merging all the end places of these networks. ALT structure allows the validation of a single condition at a time. ALT structure comprises a set of transitions equal to the number of networks to be composed alternately. These transitions are from the same input place P0. They allow, through the conditions associated with them, without ambiguity to initialize a single PN from the n modeled, which guarantees the absence of actual conflict (Figure 3). More details are presented in [20].

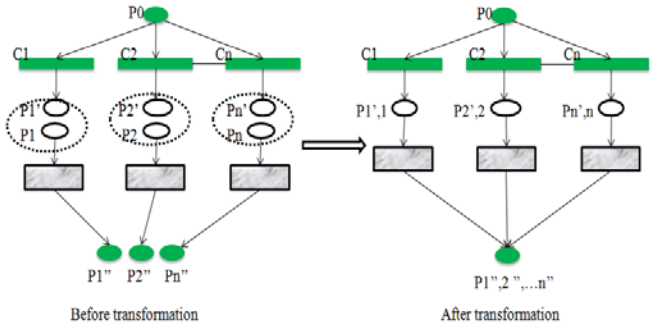

Figure 3. Alternative composition

The elementary structures represent our Meta-model. All these structures are defined manually and stored in a database.

B. Proposed approach

At a given moment, the marking of (i) the three PN of the context <user, platform, environment> and (ii) the PN of the user task, give the state of the ubiquitous HCS. The values of these markings are previously identified analyzed and stored in a database. So, this database will contain pairs of (context, task). At any time, if the values of the PN marking, describing the current context, are already included in this database, then this will be considered as an expected and well known situation and the user task will be identified, otherwise it will be considered as an unexpected situation. Managing unexpected situation will be the subject of our future research.

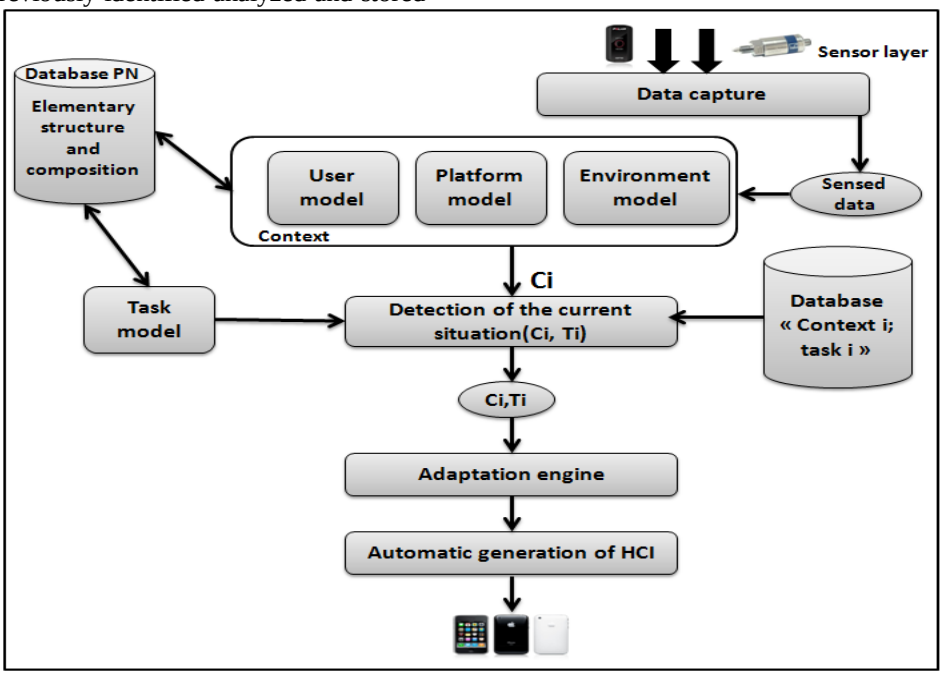

Figure 4. Proposed approach

Figure 4 illustrates our approach stating that once the data is collected from the sensor layer, it will be modeled and decomposed using PN in a user model, a platform model and an environment model. The user's task will be modeled using PN. This modeling is realized using the database of PN which contains the elementary structures and the compositions. All the (context, task) couples will be identified and stored in the "database Context i; Task i". "Context, task" database (Figure 5) is composed by two tables: context and task, connected by the association context-task. Figure 6 describes very summarily the logic diagram of database of PN. The PN is composed by elementary structures. This diagram will lead to the building of the associated database schema.

At a specific moment, the marking of the PN modeling the context will determine its current state Ci. In order to know the proper task Ti, it is required to browse the database of "context, task".

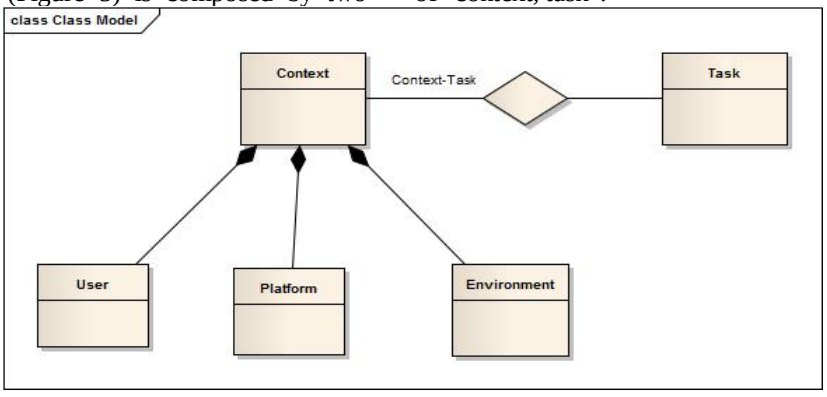

Figure 5. Logic schema of "Context-Task"

 ISBN: 978-1-61208-353-7

Whenever, the detection of the current context is made, the couple of values are transmitted to the adaptation engine. If its value is null, the adaptation engine will launch the script to deal with unexpected situation. Otherwise, it will trigger scripts to adapt to a "Known" situation. Finally, the adaptation engine will activate the automatic generation of the user interface adapted to the current context.

To manage an unexpected situation, the user will be given prior studied information and will intervene manually on the system. Actually, this is a very challenging problem that we will tackle it in our future work.

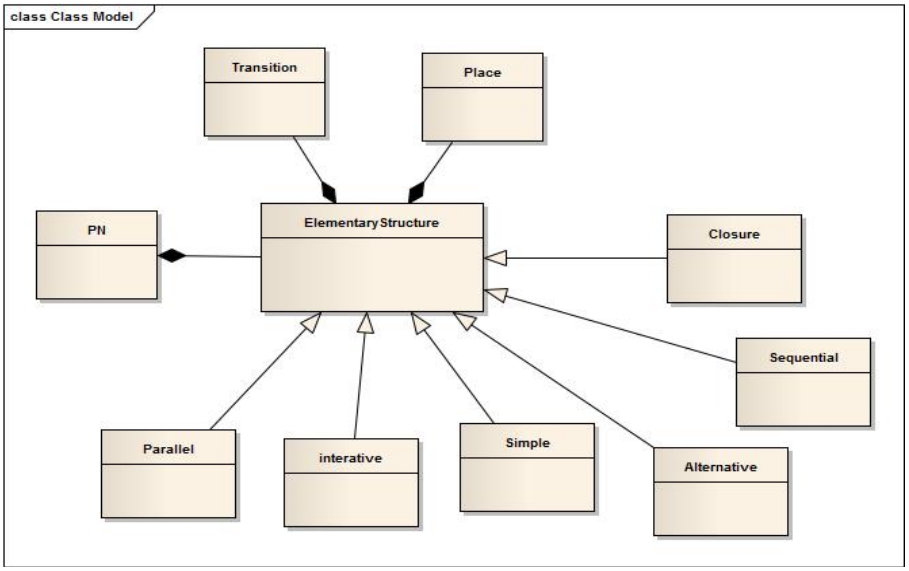

Figure 6. Logic schema of PN database

Our approach presents several advantages: first, it includes the five layers of a context-aware system, namely, context acquisition, interpretation, storage, diffusion and application layer. It separates the acquisition and modeling of context from its use. Each component of our architecture fulfills a particular task. Second, due to the complexity of context data, we choose to decompose this data into three models and to consider the user's task at the stage of modeling context. The originality of our approach lies in the proposal of the couple (context task). Indeed, the HCS became context aware and according to the context in which the user operates, the user's task may vary. In fact, each user's task is specific to a given context. That is why a set of pairs (context, task) were defined to compose the model of the pervasive HCS. Our model describes the behavior of contextual information. It decomposes the context's components into small granularity to ensure the validity of the model. To do this, we use a set of "well-organized" elementary PN structures.

Pervasive application presents a high level of dynamism so that many actions must be done in parallel. In our opinion, the aspect of parallelism in PN is very important especially in a critical domain. Certain situation requires the intervention of two or more users at the same time to meet a particular circumstance. The use of formal method to describe the behavior of a context-aware system allows us to deduct the properties of the system and the users' requirements in order to generate the appropriate interface at a given moment.

Context-aware approaches for user interface generation still have serious difficulties to dynamically and automatically adapt and generate interfaces, meeting users' requirements. These approaches are not formal and do not cover the validation of the user interface. We try to fill these gaps by proposing elements of solutions for:

Automatic deduction of user requirements: the database "Context i, Task i" is responsible to identify the appropriate task meeting the values of context.

Automatic adaptation of user interface to the context: The context changes are managed automatically by the "detection of the current situation" component. It can be considered as a representation of the smart environment.

Validation of the user interface: Thanks to the PN modeling approach, the modeled system and the generated interfaces verify the main PN properties as reachability, boundedness, liveness, etc. This guaranties the validity of the generated interfaces

Automatic generation of graphical interfaces: The adaptation engine component assumes the automatic generation of the user interface by identifying the most suitable widget to meet the user needs [21].

Comparing to the proposed approaches, seen in the second section, our architecture is centered on the context modeling and the generated user interface. The choice of the used approach for modeling context is very important. The information of context has a direct impact on the generated interface especially in the critical domains which justifies the use of formal methods in our approach.

 ISBN: 978-1-61208-353-7

The question that arises at this stage is: "how can we deduce the users' requirements from the context and the task model?"

C. Deduction of user's requirements

In a ubiquitous environment, the context model will trigger the appropriate task model. Indeed, the task depends on the context, and it is not a fixed model. Furthermore, according to the values of the context at a given moment, we can deduce the appropriate user's task. The proposed PN for context and user's task modeling is an Interpreted Petri Net IPN (Figure 7) defined by the set: < P, T, E, OB, Pre, Post, μ, Precond, Action, Info-Transition > where:

- P= set of places = {P1, P2,..., Pn},
- T = set of transitions = {T1, T2,..., Tm},
- E = set of events including the event "always present" <e>,
- OB = set of graphical objects of the interface,
- Pre: P x T → N defines the weight of the bow joining a place pi of P to a transition tk of T,
- Post: P x T → N defines the weight of the bow joining a transition tk of T to a place pi of P,
- μ: T → E associates to each transition the appropriate triggering event,
- Precond: T → Boolean Expression defines the necessary passing condition for each transition,
- Action: T → A defines the eventual and appropriate action procedure associated to each transition,
- Info-Transition: T → OB associates to each transition, the appropriate interface objects [13].

This type of networks introduces the notions of event, condition and the notion of action. Indeed, a passing condition (Cj), a trigger event (Evj) and a potential action (Aj) are associated with each transition Tj of an IPN.

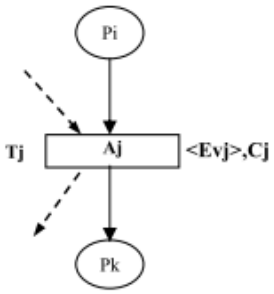

Figure 7. Interpreted Petri nets

Once the behavior of the user in its context of use is modeled, users' requirements can be deducted.

For a better management of the situation, the user needs, instantly, a lot of information. This information will be transmitted to the user by different interface components (messages, values, graphics, etc.). Since these objects are related to the context of the ubiquitous environment, i.e., the contextual parameters, we must identify the appropriate set of informational parameters for each state.

Moreover, in order to perform the tasks, the user needs to adjust some parameters in order to correct an abnormal situation or abnormal information, and/or to operate in particular situations or in collaborative tasks. For that, the interface will present a set of control components through which the user can monitor the situation; the set of these control and informational parameters constitutes the user requirements. Having presented our approach, we demonstrate its feasibility using a case study in the monitoring of a diabetic patient.

V. CASE STUDY: MONITORING OF A DIABETIC PATIENT

As a first experiment of our approach, we conducted a case study of a medical system for monitoring of a diabetic patient. This example is designed to monitor at real-time the evolution status of diabetic patients in a smart hospital. This monitoring is made possible by biological sensors implanted under the skin of the patient, which periodically control the patient's glucose levels. The ubiquitous system must continuously verify the changing state of each patient, which provides guidance on any medical interventions, or otherwise may deem any intervention to be unnecessary.

One of the problems that can arise from such a case study is to know how to notify the medical team (doctor / nurse) for an urgent and immediate intervention, and how should we proceed to carry on. This intervention should take into account the status of the patient and the location of the medical team, nurse or doctor.

The ubiquitous system will therefore generate real-time user interfaces adapted to their preferences, profiles, activities and geographical location. It will guide the user to best accomplish his task, while taking into account the various constraints of the context. In such system, the intervention of the medical team must be immediate, as the risk of loss of human life can be high. User interfaces should be validated and must present relevant and reliable information. Errors at the interfaces can cause the deaths of patients.

As a first step of our approach, we must model the information of the context. We consider the context as the triplet <user, platform, environment>. Each component of this triplet is modeled by an independent PN. Those components are:

- User's PN (Figure 8.A): it aims to identify the profile of the user (doctor or nurse). The marking of the network at a given time defines the type of the connected user. The doctor can be specialist, resident or internal.
- Platform's PN: Figure 8.B describes the different platforms that can be used by users. User interface can be hosted on various platforms. For our case study, we consider that a user can connect using a tablet, a PC or a mobile phone.
- Environment's PN: it describes the different values of our environment. For our example, after the opening of the session, various sensors intercept in parallel, the glucose level (GL) of the patient, the geographical coordinates of the user and the time. Concerning the geographical data, a user can be in the hospital, in the cabinet, at home or outside (i.e.,

in a restaurant or on the road, etc.). Concerning the time, it can have three different values: morning, afternoon or evening. Tokens present in different places, will describe the state of the environment by specifying the value of time, geographical data and glucose level (Figure 8.C).The deduction of the environment's properties must be done at the same time. This later is possible through the parallel composition of PN.

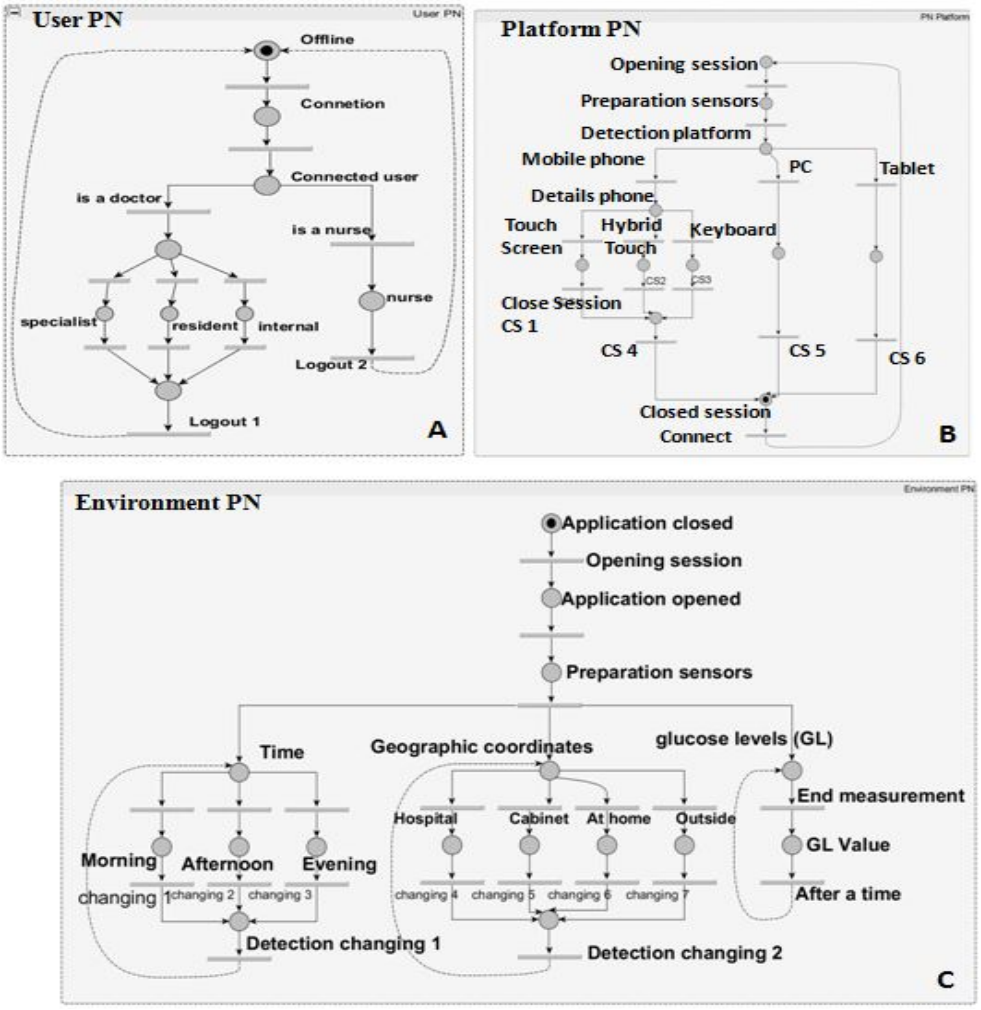

Figure 8. Context model

The environment's PN must be watchful to any changes that may happen to the environment. This action is possible by transitions "changing detection 1 and 2", which will monitor the possible changes in the environment. If any change occurs, then our sensors measuring will catch the new data. After modeling the different components of the context, we model the task.

For our example, we consider a critical situation where several actions must be done at the same time. Here is the scenario: Let us suppose that the patient is hyperglycemic (i.e., the Glucose Level >=4mmol) and the relevant doctor is not in the hospital. This situation is very critical and has to be treated by several actors. The doctor and the nurse must be aware of this critical situation at the same time, so they must perform actions in parallel. Let us notice that the system must select the most relevant replacing doctor according to his geo-location, his availability and his profile. The doctor receives a notification for an urgent and immediate intervention. In this case, the nurse cannot face alone such a situation, which requires a doctor's intervention. These two users must perform their actions at the same time. The interfaces will guide the doctor and the nurse each one according to his profile, by presenting appropriate information about the patient. The patient, suffering from a very serious condition of hyperglycemia, is in a coma. The doctor must inject insulin in him and check patient's status and measure its glucose level:

- If the rate of GL <=4 mmol, the doctor must give food containing sugar, wait 15 minutes and repeat the measurement of glucose;
- If the rate of GL > 4 mmol, the doctor must repeat the insulin injection;
- If the status is normal, then the patient's condition should be monitored for any possible change.

 ISBN: 978-1-61208-353-7

At the same time, the nurse must give plenty of water to reduce the patient's glucose. Then, she must make a urine levy. This situation is very critical and must have an immediate intervention to avoid the risk of patient's death. The actions of doctor and nurse, illustrated in Figure 9, must be done in parallel. PNs are very efficient to do this modeling.

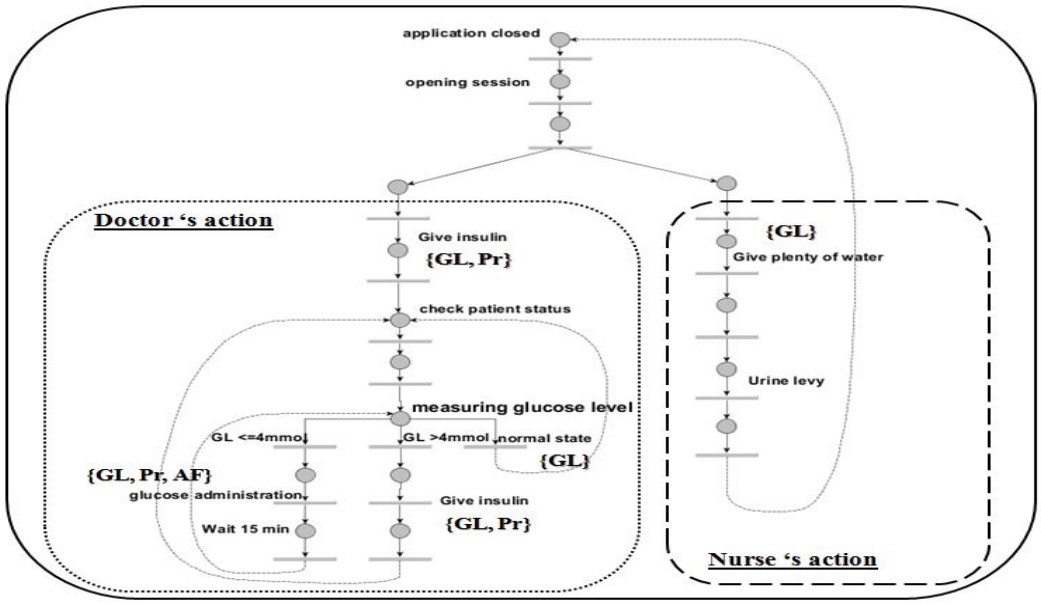

Figure 9. Task model: critical intervention on hyperglycemic patient

The values of the tokens in the context's PN trigger the appropriate Task. Concerning the deduction of user's requirements, we consider the PN transition "Begin Action". To these transitions, we associate the adequate parameter(s) of interface, either informational or control, which refer to the user's requirements. Thus, at the point of "measuring glucose level", the user disposes of the relevant user requirements in order to adequately perform their action, i.e., the Glucose Level (GL). This value comes from the context model, more precisely from the environment's PN. For instance, the informational parameter GL informs the user of the Patient's usual level of glucose. At the state "glucose administration", the user disposes of the glucose rate and other information related to the prescription of medicines (Pr). At the point of "give insulin", the user will dispose of an authorized food menu for the patient (AF) to select from. At the point of "Give plenty of water"; the user disposes of patient's glucose rate.

The compositions of elementary structure offer the possibility to the user to perform multiple tasks simultaneously. All tasks can be modeled according to the compositions shown in Section 4. The principle of the elementary compositions applied in the context modeling is also applicable to the composition of several tasks.

Once the informational and the control parameters have been identified, we can deduce the necessary components of the User interface: (i) We associate an Informational graphical object to each informational parameter; (ii) We associate a Control graphical object to each control parameter. These parameters are very important because they will lead to the graphics interface components. This interface will guide each kind of user throughout his intervention. In critical situations, user interface will play crucial role especially since the physician is not the patient's treating doctor. So he does not know the patient information. These graphic components will adapt depending on the doctor's context and profile. If he is a specialist, then no need to provide all the information and if he is an internal then the interface must provide the maximum of information to reduce the risk of errors.

For the implementation of our approach, we use a SOA (Service Oriented Architecture) [22]. First, we transform our PN model into a PNML representation (Petri Net Markup Language [23]). PNML is PN XML-based standard. Its main scope is to facilitate information exchange between PN models. Each PN is considered as a labeled graph. It contains a set of objects: places, transitions and arcs. Each object has a unique identifier and a set of labels (annotations and attributes) representing the name of a place, the inscription of a transition, etc. [23]. Algorithms are written in Java to parse the PNML file of context in order to detect the location of the tokens and consequently the current context. Once the triplet of context identified through the xml code, we browse the Mysql database "context i, Task i" in order to identify the appropriate task for the current situation. We use JavaScript Object Notation (JSON) web service [24] to query the database and to extract the name of the appropriate task. The monitoring of diabetic patient is developed using Android.

Our methodology applies to all types of applications. In fact, pervasive HCS comprises a triplet: system, task and context. Each component will be modeled using PN. The

 ISBN: 978-1-61208-353-7

designer has to analyze the system and deduce all relevant context information. Each component of the context such as user, environment and platform as well as its behavior, must be known in advance. All context values are stored in a database in which we can identify the appropriate user's task. The sensor layer will guide us on the value of the context.

VI. CONCLUSION AND FUTURE WORK

This paper presented an approach for context and task modeling based on Petri nets. We model the pervasive Human-Computer System using a composing process of elementary PN in order to verify the relevant properties of the system before the generation of the interfaces. In this paper, we have tried to demonstrate the context influence over the user's task modeling in a critical domain and the strength of PN in critical system's modeling. We also explained how to deduce the users' requirements through the task model. This formal approach is illustrated by a case study on the monitoring of diabetic patients in a smart hospital.

In the near future, we plan to address the following issues: (i) we will explain the running of each component of our approach namely the implementation of the adaptation engine and the automatic generation of user interface; (ii) we will explain how acquisition context layer will supply the model of context. We are now working on human-centred evaluations. We create an experimental platform implementing a realistic scenario that volunteer doctors and nurses will use in their work.

REFERENCES

[1] M. Wurdel, S. Propp, and P. Forbrig,"HCI-Task Models and Smart Environments", Human-Computer Interaction Symposium IFIP International Federation for Information Processing, vol. 272, 2008, pp. 21-32.

[2] A.K. Dey, D. Salber, M. Futakawa, and G. D. Abowd,"An Architecture To Support Context-Aware Applications", GVU Technical Reports, 1999.

[3] G. Calvary, A. Demeure, J. Coutaz, and O. Dâassi, "Adaptation des Interfaces Homme-Machine à leur contexte d'usage", Revue d'intelligence artificielle, vol. 18, no. 4, 2004, pp. 577-606.

[4] M. Knappmeyer, S. L. Kiani, E. S. Reetz, N. Baker, and R. Tonjes, "Survey of Context Provisioning Middleware", IEEE Communications Surveys & Tutorials, 2013, vol. 15, no. 3, pp. 1492–1519.

[5] T. Gu, H. K. Pung, and D. Q. Zhang, "A service-oriented middleware for building context-aware services", Network Computer Application, 2005, vol. 28, no. 1, pp. 1-18.

[6] H. Chen, T. Finin, and A. Joshi, "An Intelligent Broker Architecture for Context-Aware Systems", Adjunct Proc. of Ubicomp, 2003, pp. 183-184.

[7] T. Chaari, F. Laforest, and A. Celentano, "Adaptation in Context-Aware Pervasive Information Systems: The SECAS Project", Int. Journal on Pervasive Computing and Communications (IJPCC), vol. 3, no. 4, 2007, pp. 400-425.

[8] A. K. Dey, G. D. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications", Hum.-Comput. Interact, vol. 16, no. 2, 2001, pp. 97-166.

[9] A. Angham, A. Sabagh, and A. Al-Yasiri, "GECAF: a framework for developing context-aware pervasive systems", Computer Science - Research and Development, Springer, 2013, pp. 1-17.

[10] Q. Xue, X. Han, M. Li and M. Liu, "A Conceptual Architecture for Adaptive Human-Computer Interface of a PT Operation Platform Based on Context-Awareness", Discrete Dynamics in Nature and Society Journal, 2014, pp.1-7.

[11] J. Bauer, "Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic", In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp - The Sixth International Conference on Ubiquitous Computing, Nottingham/England , 2004.

[12] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Generating Context Management Infrastructure from High-Level Context Models", In: Industrial Track Proceedings of the 4th International Conference on Mobile Data Management (MDM), 2003, pp. 1–6.

[13] M. Moalla, "Reseaux de Petri interpértés et Grafcet", TSI de l'AFCET, vol. 4, 1985.

[14] S. Han and H. Y. Youn, "Petri net-based context modeling for context-aware systems", Artificial intelligence review, vol. 37, 2011, pp. 43-67.

[15] P. Reignier and O. Brdiczka, "Context-aware environments: from specification to implementation", Exp Syst, vol. 24, no. 5, 2007, pp. 305–320.

[16] J. L. Silva, J. C. Campos, and M. D. Harrison, "An infrastructure for experience centered agile prototyping of ambient intelligence", EICS '09: Proceedings of the 1st ACM SIGCHI symposium on engineering interactive computing systems, 2009, pp. 79–84.

[17] M. Giulio, P. Fabio, and S. Carmen, "Ctte: Support for developing and analyzing task models for interactive system design", IEEE Trans. Softw.Eng, vol. 28, no. 8, 2002, pp. 797–813.

[18] M. Wurdel, S. Propp, and P. Forbrig, "Hci-task models and smart environments", in: P. Forbrig, F. Patern, A. Pejtersen (Eds.), Human-Computer. Interaction Symposium of IFIP International Federation for Information Processing, Springer US, vol. 272, 2008, pp. 21–32.

[19] I. Riahi, M. Riahi, and F. Moussa, "Xml in formal specification, verification and generation of mobile hci", in: J. Jacko (Ed.), Human-Computer Interaction. Towards Mobile and Intelligent Interaction Environments, vol. 6763 of Lecture Notes in Computer Science, Springer BerlinHeidelberg, 2011, pp. 92–100.

[20] I. Riahi, F. Moussa, and M. Riahi, "Petri Nets context modeling for the pervasive Human-Computer Interfaces", Eighth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'13), Lecture Notes in Computer Science, vol. 8175, 2013, pp. 316-329.

[21] F. Moussa, I. Ismail, and M. Jarraya, "Towards a Runtime Evolutionary Model of User-Adapted Interaction in a Ubiquitous Environment: The RADEM Formal Model", Cognition, Technology & Work, Springer, 2014.

[22] E. Newcomer and G. Lomow, "Understanding SOA with Web Services (Independent Technology Guides", Addison-Wesley Professional, 2004.

[23] L. M. Hillah, E. Kindler, F. Kordon, L. Petrucci, and N. Trèves, "A primer on the Petri Net Markup Language and ISO/IEC 15909-2", Petri Net Newsletter 2009.

[24] C. J. Ihrig, "JavaScript Object Notation", Pro Node.js for Developers, 2013, pp. 263-270.

 ISBN: 978-1-61208-353-7