A Model Driven Approach For Adaptive User Interfaces Specification: User, Task and Environment Impact

Hajer Taktak, Ines Riahi, Faouzi Moussa
CRISTAL Laboratory, National School of Computer Sciences, Manouba University, Tunisia
Email: taktakhajer@gmail.com, ines.riahi@yahoo.fr, faouzimoussa@gmail.com

Abstract— **The development of ubiquitous applications is inherently complex. The adaptation process enhances the efficiency of an application by the user interface design based on several essential steps such as: system modeling, task analysis, user profiling and interface specification. A number of different approaches have been proposed to build context-aware user interfaces. In this literature, the Human-Computer interaction, namely, the task analysis is often managed without explicitly integrating the user model. This latter is operated inside a context model and has few impacts on the interaction during its progress. In order to deal with this situation, we propose an adaptive user interface based on a model driven architecture that would transform an initial task model to an interaction model through the integration of environment information, user profile and tasks to be performed. The proposed approach takes advantages of the ontology written in Web Ontology Language (OWL) in order to define user's profile and of domain ontology for persons suffering from disabilities. The abstract task model is formalized as a Petri Net that analyzes the communication flows between users, environment and tasks.**

Keywords-Human-computer interaction; user model; task model; Petri Net; MDA; TuneIn.

I. INTRODUCTION

"Ubiquitous computing is the method of enhancing computer use by making many computers available throughout the physical environment, making them effectively invisible to the user" [1]. According to this definition, the pervasiveness is guaranteed by defining dynamic users and environmental data. The user interfaces should be able to react according to the heterogeneous environment and the different users' characteristics.

Human-computer interaction, with increasingly more complex environments and user capabilities, has become a major issue. What made this issue more challenging is the emergence of the ubiquitous computing leading to new forms of interactions. The purpose is to know which information is the most appropriate for a specific user in the current environment and to improve the interaction user-system quality in such smart environments [2] [3].

The user interfaces are commonly deduced from a task model that defines elementary actions and user's activities. Indeed, it is the best way to ensures effectiveness of the application that would allow users to reach their goals and perform their tasks [4]. However, from the user task model, it is hard to provide enough information about the user abilities, preferences, goals and activities. In addition, this model does not exhibit the impact of the environment on the user system. For example, it is impossible to specify if the user has a handicap that prohibits the standard interaction mode. Therefore, during the user interface design process, it is important to consider all relevant data involved in the interfaces personalization. In order to address this issue, the proposed approach is based on how a design method is able to explicitly specify the link between the task and the user and to generate personalized contents of the user interfaces. The aim is to achieve a Human-Computer system where the interaction would become aware of contextual variations depending on the user and/or the environment in which they perform.

This paper is organized as follows: Section 2 introduces the state of the art of interface specification researches. In Section 3, we present the user-task interaction model based on a user profile allowing the creation of applications suiting different user needs and a task model able to gather information and to deduce the user interface elements. We then demonstrate the applicability of the system referring to a case study for different persons operating in several environments.

II. RELATED WORKS

The researches on user interface specification have focused on context, user and task modeling. This section presents a description of different methods for creating context-aware architectures.

Many works have been interested in the adaptive interface design. In the table I, we present a brief review of the literature on conceptual modeling of computer applications and adaptive interfaces. Through this study, we are going to analyze the existing solutions and examine the complexity, tooling and adaptation process associated with them. We have deduced comparison criteria to determine the advantages and disadvantages of each approach in the related works. The framework CAMELEON [5] used the User Interface extended Markup Language (UsiXML), one of the most advanced Human-computer interaction (HCI) approaches. User interfaces were defined at a high abstraction level. This tool is compatible with a Model Driven Architecture (MDA) [19] and exploits user and environment ontologies. Nevertheless, it does not provide a mechanism for interface validation. This gives rise to additional efforts of the designers. Limbourg *et al.* proposed a graph transformation based approach. It did not discuss the context model and had been only interested in presentation adaptation [6]. Vanderdonckt proposed a method for developing user interfaces based on MDE. His approach has been focused on presentation, content and data adaptation [7]. Therefore, the context model is not well defined and does not use ontologies while defining the user profile. The proposed approach should be able to identify the activity of the user, so as to provide the required contextual information at run and design time. Besides, the adaptation needs to focus on user's general and specific information (preferences, abilities, physical condition, age, gender, etc.). It also requires the verification of the accuracy and the reliability through an interface validation process.

TABLE I. MEETING OF IDENTIFIED CRITERIA IN SOME RELATED WORKS

	task consideration	**Context Model**		**Tooling**	**User interface validation**	**Ontology use (explicit user representation)**	**Interface adaptation**	**Adaptation way**
		User	**Environment**					
CAMELEON[5]	Yes	User ontology	Environment ontology	UsiXML	No	Yes	Presentation and content adaptation	Automatic/Manual
TOMATO[6]	Yes	Context Model		TOMATO-L Prolog	No	No	Presentation adaptation	Semi-automatic
MDE approach[7]	Yes	Context model		UsiXML	No	No	Presentation and content	--
Hachani *et al.*[8]	Yes	Context model		EMF/ ATL	No	No	Content adaptation	Semi-automatic
COMODE[9]	No	Context model based on OWL Ontologies		EMF/ OCL2.0/ UML	No	Yes	Content Adaptation	--
Bacha *et al.*[10]	Yes	User Profile	Location Time Frame External events	BPMN/ UIML/ Ontology	No	Yes	Content adaptation	Semi-automatic
Riahi *et al.*[4]	Yes	Knowledge base		Petri Net/ PNML	Yes	No	Content adaptation	Semi-automatic

Some architectures do not consider an explicit interaction between users and task and do not concentrate enough on the information provided by users, especially their disabilities. The majority of literature works are interested in context-aware interfaces generation regardless of user's task and/or user's profile. Other works have defined ontologies for user and environment with restricted data. For example, Bacha [10] focused on the information that should be provided in each situation without considering the design elements and the user requirements. He also restricted the user's description by just identifying few criteria considered as relevant (e.g., user identification, user demographics data). Nevertheless, by linking a domain ontology to a user profile ontology, we would have dynamic interface changes considering as much user's properties (e.g., cognitive, sensorial and physical abilities, activities, etc.) as possible. Bacha based his approach on a context model to generate semi-automatic user interfaces for only two methods; automatic form filling and query enhancement. This approach lacks an interface validation process.

A complementary study has been done about context-aware approaches. The majority of literature works are interested in context-aware interfaces generation without user task and/or user profile's consideration. These applications are able to gather, manage, evaluate and disseminate context information [11][12][13]. For these platforms, none of them fully satisfies an explicit interaction between the user and task models able to dynamically generate the user interface.

While defining an adaptive user interface, the task model is useful to:

- Assess the task complexity in terms of perception, analysis, decision and motor action of users in order to reach a goal.
- Describe existing systems in order to understand the design and analyze the restrictions and the way to overcome them.
- Analyze how users think the activities should be performed.

Hierarchical Task Analysis (HTA) [14], Concur Task Trees (CTT) [15], Business Process Model and Notation (BPMN) [16] and Hamsters [17] provide a particular set of elements that would be useful especially for a specific type of systems and users.

By analyzing different related works, we have noticed that systems usually have difficulties while assisting the user in performing the task. Indeed, the user's knowledge about the domain application, the display of type of preferences, and of the tasks and willingness to interact with the system are some of the elements that can be managed in a user model. They could significantly impact the task model and thus the interface generation [18].

After a careful review of definitions referring to user and task models, a key factor has been deduced so as to conduct the software development process. In order to define the task model and the transformation rules, we propose a model driven approach. The Model Driven Architecture (MDA) of the Object Management Group (OMG) represents an example of the Model Driven Engineering (MDE) that is a software development approach family based on the use of models in the software construction [19]. MDA uses models in various steps of software development cycle. It recommends the elaboration of (i) the Computation Independent Model (CIM); (ii) Platform Independent Model (PIM) and (iii) Platform Specific Model (PSM). In fact, diverse aspects of the business process and the supporting software system are captured in models and are automatically transformed to the source code of a desired platform [20].

Using this concept of MDA, the system should transform task models formalized as CTT, BPMN or Petri Net in order to provide interaction ones. An adequate mapping should be done between the CIM, PIM and PSM levels.

Following our advanced literature review regarding the model transformations, we have decided to use Petri Net formal language [4] for task modeling as it is enforceable and has many techniques for an automatic verification of interface properties (boundedness, liveness ,etc.). We also justify the use of Petri Net by its ability to guarantee the validity of the interface. In fact, the essential reason behind this choice is that the Petri Nets-based models can reliably describe the aspects of concurrency, parallelism and dynamism which are fundamental features of ubiquitous environment. A domain ontology and a profile have also been used as elements that improve the interface personalization process.

III. USER-TASK INTERACTION MODEL

Reviewing the literature solutions, we have identified three main entities in the domain of adaptive user interfaces: *user, task* and *environment*. The major challenge is to get an interaction model according to the entities previously cited. Our interests have been focused on the user and task models impact on an adaptive user interface.

Indeed, the user affects the task model to dynamically modify the interaction model depending on his profile features such as disabilities or his environment. Let us consider an example of the radio application “TuneIn”, a mobile application offering the ability to listen to streaming audio of many radio stations worldwide [21]. Basically, “TuneIn” is not aware of the environment and the user’s changes. It presents an interface which is unable to modify its contents or change its behavior. In Figure 1, we present screenshots of the current application where the car mode provides a simplified user-interface to have a quick access to the application's most used features. This functionality does not sufficiently satisfy the context awareness in terms of environment, user and task’s auto-detection. In fact, as shown in Figure 1, while the user is driving, the use of the car mode icon is obligatory. The vocal command requires a click on the search icon. Even if the environment conditions are not in favor to physically manipulate the interface, the user must interact with buttons. Thus, as described previously, the voice recognition is not automatic and the user should specify his context of use (driving) and his interaction mode. This leads to undesirable manipulation of an interface since the operation is potentially dangerous.

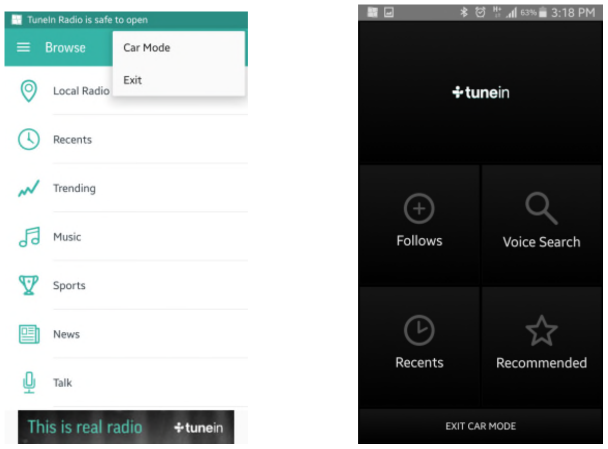


Figure 1. TuneIn Screenshots

The proposed improvements could concern user’s interest and preferences (playlist, song’s language, etc.) as well as user’s capabilities. Therefore, based on the user profile, a blind person can interact with “TuneIn” exclusively through the voice recognition. Our research aims to make adaptive user interfaces detect the user and the environmental changes through user’s profile and sensors (i.e. accelerometer).

As seen in Figure 2, the user model, combined with the environment and the task model, is able to generate a different task model (user task interaction model). To characterize this new model, we emphasize each environment parameter and the corresponding user and task.

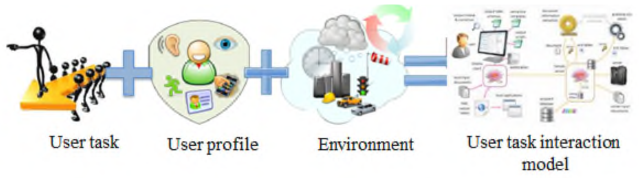


Figure 2. The user task interaction model according to the user and the environment

The user task interaction model is an alteration of elementary actions depending on users and the environment (Figure2). Indeed, any change that occurs at one of these models will impact the actions to be performed on the interfaces.

The key features of our approach are:

- Give a general and explicit representation of the user when specifying the interface.
- Get user information from a profile ontology strongly linked to domain ontology (e.g., person diseases/disabilities) for better understanding and easier interface modeling in a specific domain.
- Get generic interfaces workable by any operator. The same user interface can be exploited by a normal person and a person with disabilities. Thus,

we ensure the ability to preserve usability across multiple contexts of use [22].
- Exploit the strength and the advantages of an MDA approach to define the interaction model.
- Conceptualize the context in user interface design.
- Improve the usability and simplify the creation process.
- Reduce the complexity and ensure the validity of the interfaces.
- Provide the needed information timely and properly.
- Avoid the development of each system (or system type) in its own way with no common architecture currently available.
- Provide a set of models ensuring a dynamic interface generation depending on the user, the task and the environment in which he acts.

Thus, this work ensures the built of a standard architecture offering genericity and flexibility in smart environment. There is a need to dynamically create models that could be supported by a specific person in such environment. In order to do so, we propose an automatic model manipulation based on MDA transformation. We have been especially interested by the CIM to PIM transformation. The interaction model is deduced from the CIM representing the abstract task without showing the system structures' details. It focuses on the elementary actions to perform. The relevant information of the user and the environment is specified while defining the transformation rules. Our method should be able to automatically translate the CIM level to a PIM covering functional and environmental aspects.

Through the transformation rules, the user-task interaction model is defined at the PIM level. The interaction model is complete, generic and understandable. It includes the different information from the user, environment and task. The proposed architecture resumes our interaction model as shown in Figure 3:

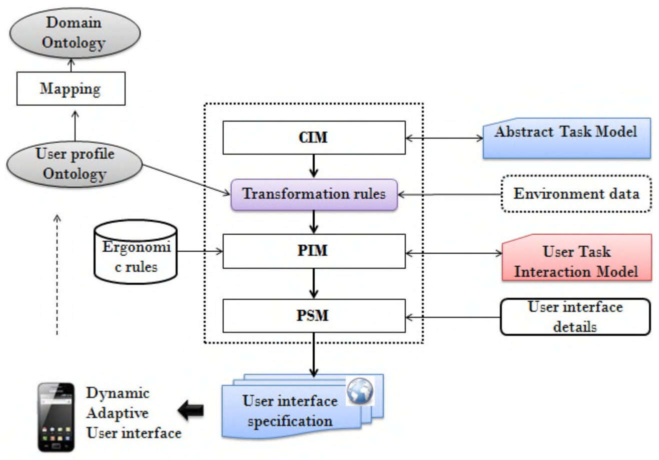


Figure 3. The proposed approach

In this architecture, we have focused on how the user actions can be influenced by the profile in a ubiquitous environment. The context is composed of static and profiled information considered as high persistence information. They are grouped in the profile ontology that communicates with domain ontology for better precision. This data is defined by the user while using his smartphone. Sensed context corresponds to information captured from sensors. In our case, this data constitutes the environment. This aspect is not addressed in this paper.

The user profile ontology is mapped to the domain ontology while the environmental data is captured through sensors. During the MDA model transformation, the PIM level holds a Petri Net Markup language (PNML) file enhanced by informational variables and user requirements. Besides, each graphic component is associated to informational and command variables.

As seen in Figure 4, the transformation consists in creating a target model from a source model by rules that describe how one or more constructs from the source model should be replaced by one or more constructs in the target model [19]. The transformation exploited in this work is an exogenous transformation meaning that the mapping of the models is written in different domain specific language [20] (as shown in Figure 4).

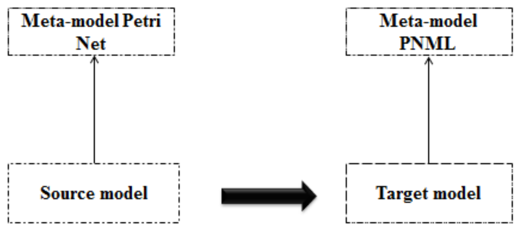


Figure 4. Exogenous transformation

To specify the source meta-model used in our approach, we have involved the meta-model proposed by our research team which consists in a process modeling based on elementary structures. This guarantees a prior validation of interfaces and saves considerable time in the development cycle of the user interface. In Figure 5, all the user's actions and components context behavior (elementary or composed) are sorted according to typical compositions: sequential, parallel, alternative, choice, iterative or of-closure. We present below, the meta-model of this process.

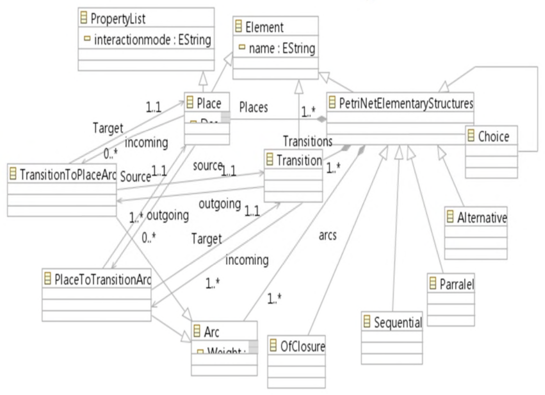


Figure 5. PetriNet Meta-Model

The target model is a PNML file aware of transformations' changes. In fact, the choice of the PNML is explained by its reliability, universality and mutuality. The main idea of PNML is that any kind of Petri net can be considered to be a labeled graph. In particular, all information that is specific to a particular kind of Petri net can be captured in labels. In the CIM to PIM transformation, Petri Net is used as an input to be transformed depending on the user profile. This step will result in a PNML file which preserves the validation features of Petri Net. Then, we use this PNML file to support the transition to a user interface markup language (UIML) to generate the interface. By this phase, we are able not only to treat context parameters affecting the user interface behavior but also to keep using the benefits of Petri Net modulation especially its validity when moving from graphical model to a markup languages. The PNML meta-model is illustrated in the Figure 6:

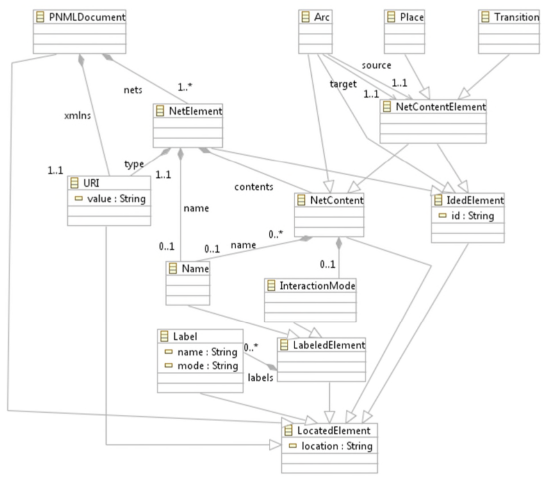


Figure 6. PNML Meta-Model

We define the transformation rules through three different levels as shown in Figure 7:

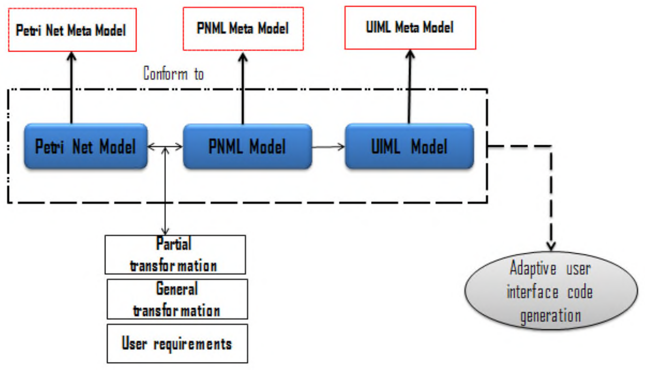


Figure 7. Transformation rules between CIM and PIM levels

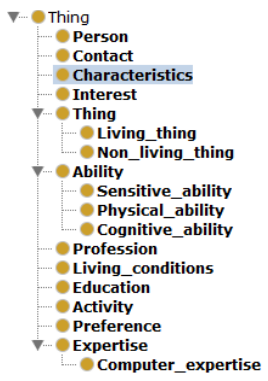
The partial level concerns the transformation of some informational data of the model (e.g., the kind of music to be streamed). The general level deals with the transformation of the entire model (e.g., voice recognition). The user requirements level is based on the identification and the modification of the user's requirements according to the current situation (e.g., buttons, size, color). The user task parameters such as control and informational variables constitute the user requirements. They are mandatory while specifying an interface managing different types of context.

IV. CASE STUDY

For application and evaluation, we have conducted a case study with a view of the radio application "TuneIn" presented in Section III. The proposed model has improved the interface behavior and has allowed the combination of several entities aspects depending on the context of use (e.g., user abilities). We have modeled and defined different scenarios in order to get personalized interfaces that respond to environmental and requirements changes. The main purpose is to enable the application to adjust to its context of use and to acquire as much data. The first user is a blind person; he manipulates the interface only by voice recognition. The second person suffers from a visual impairment; he cannot see application contents properly. The third one is a normal person being attracted by rock music. If the device is used outdoor and that the user is driving, we may experience another interaction process model.

One of the problems rising from this case study is how to dynamically adapt the interface to the user profile and the environment in which he operates. The first step of our approach is the user information modeling. The architecture shows three main components. Those components are:

- The user profile (Figure 8): It describes the user information modeling. We have defined a general, yet extendable ontology able to adapt to the needs of every application, maintaining at the same time a general common structure so as to satisfy portability and communication between different applications [18]. The used ontology is mostly static and permanent. More dynamic information is captured from the environment. The figure below defines the user profile ontology upper level classes as defined in Protégé. We have used the information introduced by the user concerning his disability (blind, nearsighted, normal) and his preferences (playlist, song's language, etc.).



- Thing
  - Person
  - Contact
  - Characteristics
  - Interest
  - Thing
    - Living_thing
    - Non_living_thing
  - Ability
    - Sensitive_ability
    - Physical_ability
    - Cognitive_ability
  - Profession
  - Living_conditions
  - Education
  - Activity
  - Preference
  - Expertise
    - Computer_expertise

Figure 8. User Profile Ontology

- The domain ontology: It allows the retrieval of the diseases' characteristics (e.g., diseases types, symptoms, causes and treatment). We have used the Generic Human Disease Ontology (GHDO) [23] in order to deduce actions to operate. In fact, GHDO defines what the person's level of functioning is and what the needs of persons with various levels of disability, impairments, activity limitations and participation restrictions are. It defines the domain that helps us to describe changes in body function and structure, what a person with a health condition can do in a standard environment (his capacity level).
- The environment: it describes the different values captured via sensors. For our example, the environment can be geographical data (position), accelerometer, etc.

In order to change "TuneIn" interface behavior, we have specified the CIM to PIM transformation. The CIM level receives the abstract task. This latter is defined as a task with no specific type. It is useful to define a process at an abstract level as shown in Figure 9.

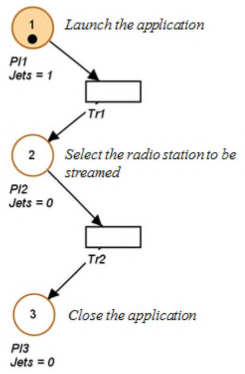


Figure 9. The abstract task Petri Net of TuneIn application

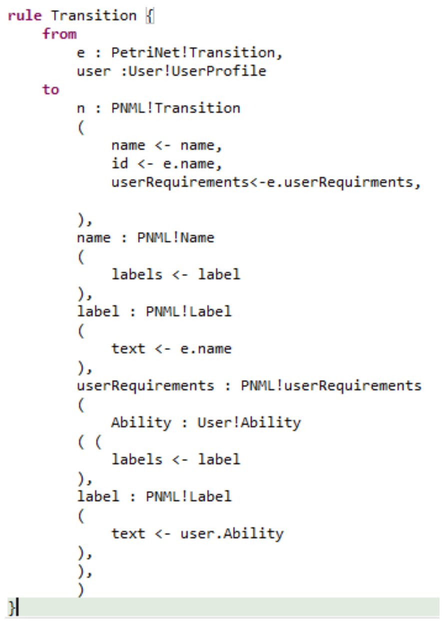
According to the abstract task and the defined rules, we deduce the new user task interaction model. By identifying the transformation rules, three new interaction models appear according to every occurred change.

Partial transformation: for a person interested by rock music, the system computes partial transformation meaning that some elementary structures are extracted from the initial Petri Net. A normal person that defines his interests (rock music) should have a restriction in the information seen in the interface. The context has the ability to change the interaction process by deleting some entities and keeping only the most significant ones. The environment information captured through sensors (accelerometer) shows that, when the user is driving, then a voice command is needed.

User requirements transformation: For a person having a binocular acuity of distant vision, the system communicates with the domain ontology and decides which are the most suitable representation and interface objects' interaction of the according users' needs. In this case, in order to generate an interface related to a nearsighted person (big characters), we should specify the size of variable information and instructions (text boxes, buttons, labels, etc.) The variable output size switches from 12 to 18 according to the user's disability.

General transformation: For a blind person, the user capabilities will influence the generation of the interaction process model. To know about these changes, it is necessary to get user capabilities information through profiles. We have used domain ontology (user abilities) from which we deduce static characteristics of a disease. For this transformation, we do not need to analyze environmental data because the only way to interact with the interface is voice command.

Atlas Transformation Language (ATL) is applied in the context of the transformations previously cited [24]. The case study of the blind person requires a transformation program that converts an initial Petri Net model to a PNML file computing data through voice recognition. Figure 10 shows some parts of the ATL transformation.



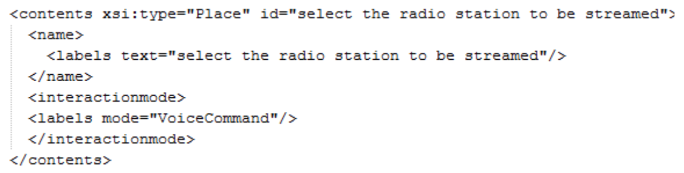
```
rule Transition {
    from
        e : PetriNet!Transition,
        user :User!UserProfile
    to
        n : PNML!Transition
        (
            name <- name,
            id <- e.name,
            userRequirements<-e.userRequirments,

        ),
        name : PNML!Name
        (
            labels <- label
        ),
        label : PNML!Label
        (
            text <- e.name
        ),
        userRequirements : PNML!userRequirements
        (
            Ability : User!Ability
        ( (
            labels <- label
        ),
        label : PNML!Label
        (
            text <- user.Ability
        ),
        ),
        )
}
```

Figure 10. An excerpt from ATL transformation rules

The "transition" rule generates tags according to the user requirements that constitute the crossing conditions from one action to another. The "place" rule generates a PNML file that contains the interaction mode according to the user disability (Blind) from the place elements. This is achieved by adding the appropriate tag (mode) inside the informational variables. The elements of the PNML file correspond to those generated by place, transition and arc rules. We do not need to specify user characteristics because they are directly extracted from the user profile (e.g., user.Ability).

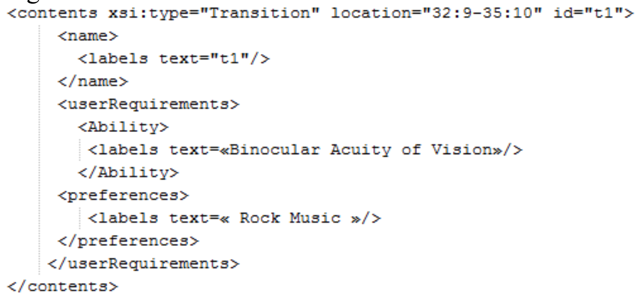
The transformation rules define how the different entities have the ability of mutually affecting each other in order to generate a contextualized user interface. The proposed model allows the combination of several entities aspects (user, task and environment). Based on this combination, we have deduced the new interaction model showing the actions to be performed on the interface. From the CIM to PIM transformation described above, we can notice that the PNML file has been extended with the interaction mode tag described in Figure11. If the system detects dynamically that the user is blind, the interaction mode switches directly to voice command.



```
<contents xsi:type="Place" id="select the radio station to be streamed">
  <name>
    <labels text="select the radio station to be streamed"/>
  </name>
  <interactionmode>
  <labels mode="VoiceCommand"/>
  </interactionmode>
</contents>
```

Figure 11. An excerpt from the extended PNML file

If the extracted user profile data concerns the user requirements (binocular acuity of vision, music preferences), they are directly defined in the PNML file as shown in Figure 12.



```
<contents xsi:type="Transition" location="32:9-35:10" id="t1">
     <name>
       <labels text="t1"/>
     </name>
     <userRequirements>
       <Ability>
        <labels text=«Binocular Acuity of Vision»/>
       </Ability>
     <preferences>
        <labels text=« Rock Music »/>
     </preferences>
    </userRequirements>
</contents>
```

Figure 12. An excerpt from the extended PNML file

The model guarantees the consideration of design specification before the implementation phase. From such an interaction process, we have kept the approach at an abstract level separating application reasoning from the implementation technology. The proposed model respects the MDA architecture to generate a code corresponding to a specific model deduced from users, environment and tasks. After defining the PIM level, we would be interested in the details specifying the system behavior on each particular platform's type [19] to finally generate a contextualized interface based on the user requirements.

V. CONCLUSION

This paper presented a MDA approach that considers user profile properties and environment information in order to generate an adaptive user interface and also to satisfy an exceptional interaction need of a disabled person. Our work transforms an initial Petri Net model to a PNML based interaction one by combining tasks, users and environment. We have focused on gathering all available user abilities and preferences, since an early design stage, in order to have as generic interface adaptation as possible. Thus, we would need to enhance the re-usability of transformation rules and complete the domain ontology integration so that we would retrieve relevant description which is able to dynamically modify the interface behavior according to its matching to the symptoms of different user's disabilities. This would build a PSM model to generate the relevant code. Furthermore, we plan to develop a critical context aware application from the modeling stage until the code generation in order to investigate runtime validation techniques. We are finally working on merging our approach with contextual web services creation.

REFERENCES

[1] M. Weiser, "Some computer science issues in ubiquitous computing," Communications of the ACM, 37(7), 1993, pp. 75-83.

[2] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, D. Riboni, "A survey of context modeling and reasoning techniques," In Pervasive and Mobile Computing, 2010, pp. 161-180.

[3] B. Schilit, N. Adams, N.L. Want, "Context-aware computing applications," In IEEE Workshop on Mobile Computing Systems and Applications, 1994, pp.85-90.

[4] I. Riahi, F. Moussa, "A formal approach for modeling context-aware human-computer system," Computers & Electrical Engineering, 2015, pp. 241–261.

[5] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, J. Vanderdonckt, "A Unifying Reference Framework for multi-target user interfaces," Interacting with Computers (IWC), 2003, pp. 289-308.

[6] Q. Limbourg, J. Vanderdonckt, "Transformational Development of User Interfaces with Graph Transformations," CADUI, 2004, pp.105-118.

[7] J. Vanderdonckt, "Model-Driven Engineering of User Interfaces: Promises, Successes, Failures, and Challenges," Proceedings of Annual Romanian Conference on Human-Computer Interaction 01/2008.

[8] S. Hachani, S. Dupuy-Chessa, A. Front, "Une approche générique pour l'adaptation dynamique des IHM au contexte," In Proc. IHM ACM Press, 2009, pp. 89-96.

[9] S. Vale, S. Hammoudi, "COMODE : A Framework for the developement of Context-aware Applications in the context of MDE," Internet and Web Applications and Services, ICIW, 2009,pp 261-266.

[10] F. Bacha, K. Oliveira, M. Abed, "A Model Driven Architecture Approach for User Interface Generation Focused on Content Personalization," In Proc.IEEE RCIS, 2011, pp. 1-6.

[11] J. Hussein, A. Colman, "An architecture based approach to context-aware adaptive software systems," Tech. rep., Faculty of Information and Communication Technologies (FICT) Swinburne University of Technology, 2011, pp. 154 - 163 .

[12] F. Buttussi, "A user-adaptive and context-aware architecture for mobile and desktop training applications," In: Proceedings of the 10th international conference on human computer interaction with mobile devices and services, MobileHCI, 2008, pp. 543–543.

[13] X. Qing, X. Han, M. Li, M. Liu, "A conceptual architecutre for adaptive human computer interface of a PT operation platform based on contet-awareness," Discret Dyn Nat Soc Journal, 2014, pp 1-7.

[14] J. Annett, "Hierarchical task analysis," Handbook of cognitive task design, 2003, pp. 17-35.

[15] F. Paternò, "ConcurTaskTrees: an engineered notation for task models," The handbook of task analysis for human-computer interaction, 2004, pp. 483-503.

[16] P. Wohed, W. van der Aalst, "On the Suitability of BPMN for Business Process Modelling," 4th International Conference on Business Process Management, 2006, pp 161-176.

[17] C. Martinie, P. Palanque, M. Winckler, “Structuring and composition mechanisms to address scalability issues in task models,” in Human-Computer Interaction– Article in a conference proceedings INTERACT, 2011, pp 589-609.

[18] M. Golemati ,A. Katifori ,C. Vassilakis ,G. Lepouras, C. Halatsis, “Creating an Ontology for the User Profile: Method and Applications,” RCIS 2007.

[19] J. Miller, J. Mukerji, "MDA Guide Version 1.0.1.," OMG, 2003.

[20] P. Hoyer, T. Kopp, S. Abeck, “A Model-Driven Development Approach Focusing Human Interaction,” Advances in Computer-Human Interactions, ACHI , 2009, pp. 90-96.

[21] https://tunein.com/get-tunein/ [accessed March 2016]

[22] V. Alvarez-Cortes, V. H. Zárate Silva, B. E. Zayas Pérez, A. R. Uresti, A. Quintero Reyes, “User and Task Models Impact on an Adaptive User Interface for the Startup of a Power Plant,” Software Engineering, 2007 ICSE,.

[23] M. Hadzic, E. Chang, “Ontology-based Support for Human Disease Study,” Proceedings of the 38th Hawaii International Conference on System Sciences HICSS, 2005, pp.143a.

[24] S. Cuadrado, J. Guerra, E. J. de Lara,“A Component Model for Model Transformations,” Software Engineering, IEEE Transactions on ,2014, pp.1042-1060.