IADIS International Journal on Computer Science and Information Systems
Vol. 6, No. 2, pp. 69-94
ISSN: 1646-3692

USING CONTEXT MODELING AND DOMAIN ONTOLOGY IN THE DESIGN OF PERSONALIZED USER INTERFACE

Firas Bacha, Káthia Oliveira and Mourad Abed
University of Valenciennes and Hainaut Cambrésis, LAMIH, F-59313 Valenciennes, France, CNRS, FRE 3304, F-59313 Valenciennes, France University of Lille North of de France

ABSTRACT

With the diversity of computer devices and the increasing number of software systems to support daily activities, personalization became an important requirement in software development. User interface of such systems should not only be customized in terms of layout, screen resolution, and other design aspects, but should also provide the user with pertinent information that takes into account the context when using the system. We argue that personalized information should guide interface design aspects as well as the choice of the relevant information to be provided through such interface. To address this idea, we explicitly define a context model to capture all relevant information related to the user. This context model is generic and could be used in the design of any user interface. To adapt such model to a specific application domain, we propose that the context model concepts are mapped to the concepts of an ontology that captures the knowledge of an application domain. In this way, it is possible to indicate which specific application domain information should be provided as input/output in the user interface based on the context. In addition, we embed the context model and the domain ontology in a model-driven architecture framework to allow semi-automatic generation of personalized user interfaces. Finally, an illustrative example of the proposed approach is presented.

KEYWORDS

Context Modeling, Personalization, User Interface Design, Ontology, Model-Driven Architecture

1. INTRODUCTION

The proliferation of electronic information available via different computer devices with several interaction modes brings a new challenge to system development: presenting the relevant information for a specific user in a suitable manner for that user (Van Setten, 2001), that is, to generate personalized information. In this way, the user feels like the software system was developed for him/her and the software system becomes more productive and

attractive to the user, which results in a better acceptation and use of the system. To address this important challenge, information about the user and his/her situation or context while using a system is usually organized in models that are integrated to the software system. We argue that this personalized information should also be used in the design of user interfaces.

Recognizing this need, some approaches emerged to consider the context in the user interface design (e.g. Calvary et al., 2003; Vanderdonckt, 2005; Taconet and Aoul, 2008; Ayed et al., 2007; Hachani et al., 2009). These approaches use information about the user and the context to set how to present information and to customize the interface for different devices. However, the personalized information should also be used to set the pertinent input/output content in the interface fields. In other words, while designing a user interface, we would like to personalize the way information will be presented (i.e., the container) and to indicate which information should be presented (i.e. the content) considering what we know about the user such as personal data, preferences, hardware s-he is using, etc. To that end, we propose to use a context model with personalized information about the user and an ontology for the specific application domain. These models are used in a model-driven architecture (MDA) approach (OMG, 2001) to support the user interface generation.

When talking about personalization, one can think of information retrieval domain methods and techniques to identify personalized information about the user such as: cognitive methods and collaborative filtering (Cinquin et al., 2002). In general, these approaches use some basic information about the user, a class of users, and/or an analysis of contextual data (historical information) to identify the preferences, profile, and characteristics of the user. Our goal is not to provide a method like these approaches, but rather to use the information generated by them in the design of user interfaces. The information related to a particular user is viewed as an instance of a general model of context. This paper presents how this context model is defined for our purposes and its association with domain ontology in order to be used during the user interface design following a MDA approach.

The remainder of this paper is organized as follows. Section 2 presents some definitions and relevant research on personalization and context modeling. Section 3 describes our proposition of context modeling and how it is associated to domain ontologies to allow personalization. Then, in section 5, these models are used in a MDA approach to generate personalized user interfaces. Section 5 and 6 present, respectively, some related works and limitations of our approach. Finally, Section 6 presents our conclusions.

2. PERSONALIZATION AND CONTEXT MODELING

There is no consensual definition of personalization. Usually, authors define personalization based on their specific goals and applications (Anli, 2006 and Bouzghoub, 2004). Some important definitions from the Human-Computer Interaction and Information Research domains are:

- “Personalization is the ability to provide content and services that are tailored to individuals based on knowledge about their preferences and behavior” (Hagen et al., 1999).
- “Personalization is the capability to customize communication based on knowledge preferences and behaviors at the time of interaction” (Dyche, 2002).
- “Personalization means delivering to a group of individuals, relevant information that is

retrieved, transformed, and/or deduced from information sources" (Won, 2002).

Besides the diversity of definitions, personalization is often confused with "*adaptation*". Ledoux (2001) defines adaptation as the process of modifying systems to work adequately in a given context, which means the system suits perfectly user expectation in a given context. Some authors (Kappel et al., 2000; Mobasher, 2000 and Ledoux, 2001) argue that personalization and adaptation are synonymous; others (Anli, 2006 and Won, 2002) consider that personalization is part of adaptation. Garía-Barrios et al. (2005) define personalization as "adaptation towards a *named user* for which an internal and individual model is needed". Simonin and Carbonell (2006) describe personalization as "the dynamic adaptation of the interface to the profile". In general, we can say that personalization deals with the capacity of adaptation of a user interface (UI) considering some information related to this user. The personalization can take into account several aspects (e.g., navigation, structure, functionalities) and it can be performed basically on the interface containers presentation; i.e., layout, colors, sizes, and other design elements, and content; i.e., data, information, document (Anli, 2006 and Brossard et al., 2007). Figure 1 illustrates examples of content and containers' personalization of a system for travel planning. For instance, the adaptation of the size of interface elements such as fonts and widgets represents a personalization of the container presentation based on the specific platform/device (in this case, IPhone) used by the user. Information about the departure city and departure date (Figure 1(a)) are examples of personalization of content. Departure city is automatically filled based on the location where the user is at the time of using the system. Departure date is the current day. Another example of personalization of content is presented in Figure 1(b). In this example the user is unable to walk; thus, the system will propose to him only direct itineraries with a reduced price (according to his/her age).

.

(a) (b)

Figure 1. Examples of user interface personalization in a transport system

Different approaches have been proposed to support personalization. Many of them are based on algorithms that recognize user behavior patterns when interacting with computing systems to predict user next move (Hirsh et al., 2000). Using the past and recent information about the user's interaction, different techniques of filtering and recommendation have been explored - see for example (Brusilovsky et al. 2007) for different studies on web personalization. Other studies focused on the definition of user profile models to perform personalization (Brusilovsky et al. 2007). Moreover, some authors (e.g. Calvary et al., 2003; Vanderdonckt, 2005, Hachani et al., 2009 and Ayed et al., 2007) proposed to use context

models to allow the user interface adaptation or container's personalization.

According to (Dey, 2001) when a system uses context to provide relevant information and/or services to the user, it is considered context-aware. This is a generalization of the first definition for context-aware computing which considered that context-aware software (Schilit et al. , 1994) "adapts according to the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes to such things over time", that is, context-ware applications adapts themselves according to the context. In this sense and based on the previous definitions of personalization, we can say that if we do personalization by using the context, we are building context-aware software system.

Context is being also largely defined. Schilit et al. (1994) introduced that the important aspects of context are: where you are, who you are with, and what resources are nearby. From this definition, several authors explored different context elements, such as location, environment, states of interest, time, activity, and so on. After analyzing several definitions of context, Abowd et al. (1999) defined context as any information that can be used to characterize the situation of an entity (person, place, or object) considered relevant to an interaction between a user and a system. Finally, in the user interface design research (our particular interest in this paper), Calvary et al. (2003) defined that context is composed of three classes of entities: the *user* of the system, the *platform* (hardware and software) that is used for interacting with the system and the *physical environment* where the interaction takes place.

The context can be modeled in different ways (Strang and Linnhoff-Popien, 2004): using a simple list of attributes with values (named Key-Value model); based on XML composed of tags; using entity-relationship models; using UML or ECORE diagrams; using fact and rules (i.e., logic based models) or ontologies.

We found 18 proposals of context modeling in literature. We classified them in three groups considering the context dimension defined by Calvary et al. (2003). The first group refers to the proposals that consider only one of the context dimensions, that are: context models that focus on *user* dimension, like profile, interests, and so on (UMO, 2003; Rousseau et al., 2004 and Kostadinov, 2008), that detail the *platforms* (FIPA, 2001) and its relation with the environment (W3C, 2009); or that focus on the *environment*, in particular considering information about the location (Becker and Dürr, 2005) and the time (Hobbs and Pan, 2006).

In the second group, we included the proposals that consider all the dimensions but are specific to a particular domain or technology which implies that they are detailed to answer their specificity. In this group, we quote: purposes for ubiquitous computing (Chen et al, 2004; Lin et al. 2005), for smart homes (Kim and Choi, 2006), for mobile applications (Weißenberg, 2004; Schmidt et al., 1999 and Korpipää et al., 2003) and for e-commerce applications (Taconet and Aoul, 2008).

Finally, in the third group we included the proposals that are domain-independent and that contain the three context dimensions (user, environment and platform), however, not well-detailed. Preuveneers et al. (2004), for example, includes the definition of user profile in user dimension, but do not identify what to consider in this profile. Wang et al (2004) and Arabshianand and Schulzrinne (2006) proposed upper-level ontologies with very general concepts (such as computing device and location/physical environment) that should be mapped to a specific domain of interest. Finally, UsiXML project (Vanderdonckt, 2005; Limbourg et al. 2005 and UsiXML, 2007) explored more the platform dimension than the other dimensions, since its main goal is to support the adaptation UI. The environment model takes into account only three aspects (light, noise and stress). The user model, initially limited

to a few attributes that describe the experience of the user with the platform, was recently modified to consider features that affect a UI (Tesoriero and Vanderdonckt, 2010). This is done in two levels of abstraction: a feature level where the designer define the user features that is relevant to the application domain; and, a profile level that characterize the features according to runtime situations. Although this meta-definition provides flexibility in defining the user dimension, it requires an extra-effort of the designer to define all the features for each new application domain.

3. OUR CONTEXT MODELING AND ONTOLOGY MAPPING

As mentioned previously, our goal is to use context modeling from the beginning of user interface design to personalize its containers and content being, therefore, context-awareness. To allow better productivity and semi-automatic generation of user interfaces from its design, it is essential to use a generic context model that can be used in the design of any interface independently of the domain, software, and hardware platforms. However, this context model should be detailed enough to make content personalization feasible, where we should take into account the information of a specific domain. The main difficulty in defining such context model is to find a compromise between *generality* and *specificity* so that the proposed model is reusable in different user interface designs, on one hand, and could be adapted for different application domains, on the other hand.

The first idea was to reuse an existing context model. However, as presented in the previous section, we concluded that the proposed context models were either very generic that would make difficult the content personalization, or too specific for a particular domain or for a single context dimension that could not be re-used in different applications. We decide, therefore, to consider the literature but to define our own context model. To represent the context model, we decided to use the *ECORE* diagrams that can be easily represented and integrated in a MDA approach for user interface generation (see Section 4).

Next sections present how we define this context model (Section 3.1) and how we use this model for content personalization (Section 3.2).

3.1 Context Modeling

To address context modeling, we analyzed all the concepts and proprieties of the 18 proposals of context modeling presented in section 2. Then, we classified them in categories of concepts according to their meaning (e.g., user name, address, phone were classified as user identification). Finally, we organize categories around the three main context dimensions proposed for UI design (presented in Section 2): user, platform and environment.

Tables 1, 2 and 3 show a summary of all concepts, their category, and their references for the environment, user, and platform dimensions, respectively. These concepts were organized either as classes or attributes of classes using *ECORE* diagram. Next sub-sections present the detailed model for user (Section 3.1.1), platform (Section 3.1.2), and environment (Section 3.1.3).

3.1.1 User Modeling

The user profile is composed of five major categories that describe the user during its interaction with the platform (Figure 2): demographic information, contact information, user preferences, user state, and user abilities and proficiencies.

In the literature, some authors (Kostadinov, 2008; Jrad et al., 2007 and UMO, 2003) separate "contact information" and "User Demographic data» and others (Rousseau et al., 2004; Lin et al., 2005 and Preuveneers et al., 2004) mix them. For clarity, we followed the first team, as follows:

- *Contact information* – contains personnel data (address, tel. number…) that can be changed;
- *Demographic information* - contains basic data of the user that usually do not change, such as date and place of birth or gender.

The Preference class represents the user's preferences and interests taken from different dimensions. Some approaches use only the term "preference" (Kostadinov, 2008; Preuveneers et al., 2004 and UMO, 2003), while others (Rousseau et al., 2004) use the term "interest". Although there is the possibility of using complex types to express preferences (preference compound, binary together ...) such as those proposed by Kostadinov (2008), we have chosen only unitary and simple preferences, composed of a single attribute with a Boolean type in order to indicate if the user likes or not the specific preference. The User State class presents the state of the user when interacting with the system. From the literature, this state may be emotional, physiological (Shmidt et al., 1999 and UMO, 2003) or may be an activity practiced by the user (Kim and Choi, 2006 and Korpipää et al., 2003). We did not include the emotional state in our model, since it is rarely used when describing the user. Finally, Ability and Proficiency class represents the user knowledge, skills (e.g., computing, well-writing) and abilities (e.g., walk, hear). This class is an adaptation of that proposed in (UMO, 2003).

Table 1. Concepts of environment dimension

Category - Concepts		**Reference**
Location	Location (IndoorSpace – OutdoorSpace)	(Wang et al, 2004)
	Geometric (GPS), Symbolic	(Becker and Dürr, 2005)
	Building { Indoors, Outdoors }, GPS Location	(Korpipää et al., 2003)
	Country, City, zip code, longitude, latitude, coordinates	(Arabshianand, 2006)
	Geographical place (street, city, province, country)	(Kim and Choi, 2006)
	Geocordinates, UTMCoordinates, WGS84Coordinates, Geographical Coordinate Reference System	(W3C, 2009)
	Relative, absolute	(Preuveneers et al., 2004)
	location (absolute position, relative position, co-location)	(Schmidt et al., 1999)
Time	Time	(Arabshianand, 2006), (Korpipää et al., 2003)
	TemporalEntity, Interval, Instant, DurationDescription, TemporalUnit	(Hobbs and Pan, 2006)
External events	Sound: Intensity { Silent, Moderate, Loud }	(Korpipää et al., 2003)
	Light: Intensity { Dark, Normal, Bright }	
	Light: Type { Artificial, Natural }	
	Light: Source Frequency { 50Hz, 60Hz, Not Available }	
	Temperature { Cold, Normal, Hot }	

	Humidity { Dry, Normal, Humid }	
	Sound: Type { Car, Elevator, Rock Music, Classical Music, Tap Water, Speech, Other Sound }	
	Temperature Value	(Taconet and Aoul, 2008), (Kim and Choi, 2006), (Preuveneers et al., 2004)
	Lighting	(Lin et al., 2005), (Preuveneers et al., 2004)
	Noise	
	IsNoisy – IsStressing – LightingLevel	(UsiXML, 2007)
	Humidity – Pressure - Environmental condition	(Preuveneers et al., 2004)
	physical conditions (noise, light, pressure, temperature ,acceleration)	(Schmidt et al., 1999)

Table 2. Concepts of User Profile Dimension

Category - Concepts		**Reference**
User identification	Family name, address, e-mail, phone/ fax number	(Kostadinov, 2008)
	Name	(Kostadinov, 2008), (Weißenberg, 2004)
	Contact Information/detail (city, country, email, family name, fax/ phone number, full name, postal code, street)	(UMO, 2003) , (Rousseau et al., 2004)
User demographic data	Date of birth, occupation, children, revenue, marital status	(Kostadinov, 2008)
	Demographics (age, age group, birthday, birthplace, salary, employment, family status, first language, gender, wealth)	(UMO, 2003)
	Gender	(Lin et al., 2005), (Kostadinov, 2008)
	Affiliation	(Rousseau et al., 2004)
Interests/ preferences	Preference (interface preference, privacy preference)	(UMO, 2003)
	Movie preference, music preference, news preference	(Kim and Choi, 2006)
	Preference profile	(Preuveneers et al., 2004)
	Simple preference, complex preference	(Weißenberg, 2004), (Kostadinov, 2008)
	Interest	(UMO, 2003), (Rousseau et al., 2004)
	Interests (Olympic, shopping, sightseeing, entertainment)	(Weißenberg, 2004)
Skills	First Language, Second Language, Knowledge, Computer skills, Reading skills, writing skills, typing skills	(UMO, 2003)
	Career	(Lin et al., 2005)
	language read – language spoken – language written	(Weißenberg, 2004)
	Competency (Skills, knowledge), Qualifications	(Rousseau et al., 2004)
	Habits	(Schmidt et al., 1999)
Level of edu-cation	Author, developer, learner, reader, teacher, user, ...	(UMO, 2003)
	Profession	(Kostadinov, 2008)
Ability / Disability	Abilities, disabilities	(Rousseau et al., 2004)
	Ability and Proficiency (ability to talk, to drive, to hear, to see …)	(UMO, 2003)
User State	Physiological State (blood pressure, injury, respiration, temperature , ...)	(UMO, 2003),

	Motion (lying, going up stairs, sitting, standing, walking)	
	Emotional state(anger, anxiety, disgust, happiness, sadness)	
	Mental State (depression – irritation – nervousness – psychopathy – trauma)	
	Emotional state - biophysiological conditions	(Schmidt, et al., 1999)
	Activity (Sleeping, Watching TV, Cleaning, Getting Up)	(Kim and Choi, 2006), (Preuveneers et al., 2004), (Weißenberg, 2004), (Rousseau et al., 2004), (Korpipää et al., 2003) (Wang et al, 2004)
	Mood	(Preuveneers et al., 2004)

Table 3. Concepts of platform dimension

Category - Concepts		**Reference**
Software	Operating system (Win Mobil, Symbian, Android)	(Taconet and Aoul, 2008)
	API, RuntimeEvironment	(W3C, 2009)
	OS (name-vendor-version)	(FIPA, 2001), (Preuveneers et al., 2004) (UsiXML, 2007)
	Software (name, edition, version), virtual machine, middleware, rendering engine, operating system	(Preuveneers et al., 2004)
Hardware	Memory, CPU	(Taconet and Aoul, 2008), (FIPA , 2001), (Preuveneers et al., 2004) (W3C, 2009)
	Connection	(Taconet and Aoul, 2008), (FIPA, 2001)
	Display (resolution)	(Taconet and Aoul, 2008)
	Keyboard type (Numeric, qwerty , Touch screen)	
	Network Interface (3G, WIFI, Bluetooth)	
	UI-screen (width-height-unit-resolution-color)	(FIPA, 2001)
	Connection (information-QOS information)	
	Network	(Lin et al., 2005), (Preuveneers et al., 2004) (Wang et al, 2004)
	Resource (Power – memory – CPU – storage – network)	(Preuveneers et al., 2004)
	File format	(Kostadinov, 2008)
	NetworkEntity (NetworkMode – NetworkSupport – NetworkTechnology)	(W3C, 2009)
	Screen Width – Screen Hight – Screen Size Char - Max Screen Char - Is image capable - Pointing device – Has Touch Screen - Storage Capacity	(UsiXML, 2007)
	Surrounding resources for computation	(Schmidt et al., 1999)

3.1.2 Platform Modeling

This model (Figure 3) is necessary since it describes the platform that the user will interact with – this is the reason we should use a generic way to characterize the platform. According

to the literature (Preuveneers et al., 2004; Kostadinov, 2008; FIPA, 2001; W3C, 2009 and UsiXML, 2007), the classical classification to describe a platform is to differentiate between hardware and software. In addition, each platform has a well-defined type (e.g., Laptop - PC - PDA) (Taconet and Aoul, 2008; Kim and Choi, 2006 and FIPA, 2001) and must have a unique identifier (e.g., the serial number).

The *hardware* part describes all platforms' physical aspect and it is composed of four subparts as defined by Taconet and Aoul (2008), FIPA (2001) and Preuveneers et al. (2004):

- *Memory* - to specify the RAM size of the platform.
- *CPU* - to represent the processor embedded in the platform and its speed. This information may be useful to know whether the target platform can execute or not the user interface.

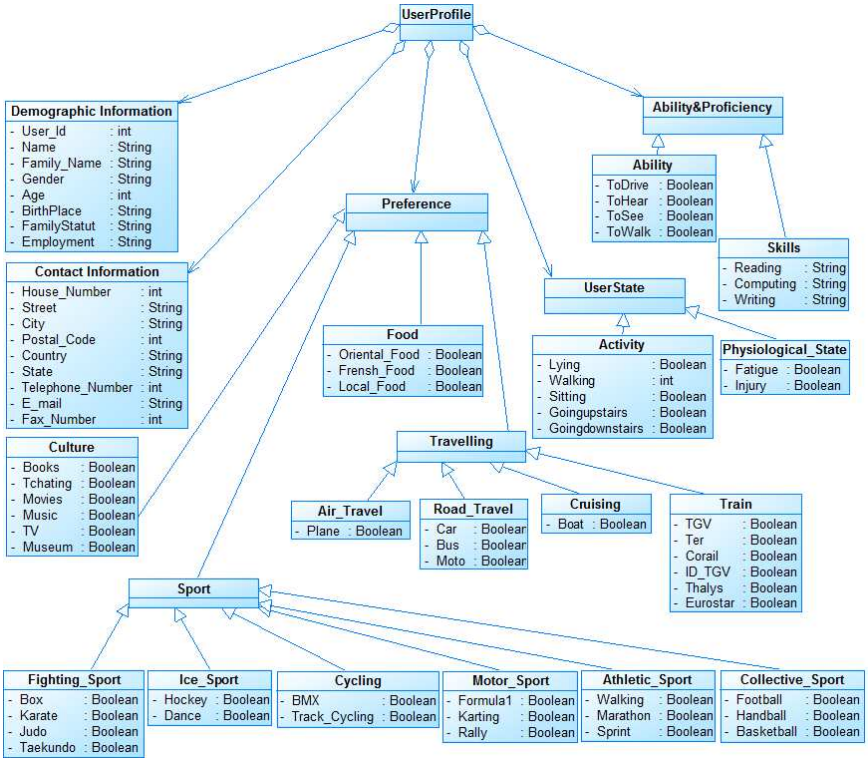

Figure 2. The user profile model

- *Network* - in order to provide general information about the characteristics of the network installed on the platform. We can exploit this information to determine if the platform has the ability to be mobile or not (in the case of Wi-Fi applications).
- *User interface* - to indicate the height and width of the user interface as well as its image resolution. The attribute "Color" is used to indicate whether this is a color interface or not. This type of feature is important to involve because the adaptation of the size of interaction interface's elements takes place according to him.

The *Software* part defines the software side of the platform and is composed of four subparts as defined by Preuveneers et al. (2004):

- *Virtual machine* - to describe the set of execution environments whose platform is equipped with. We can exploit this information in case of having a portable code (e.g. Java) where it is important to know if the platform contains the suitable VM to execute the (e.g., Java VM).
- *Application system* - to specify the set of applications installed on the platform.
- *Operating System (OS)* - to introduce the operating system that the platform works with. Such information is essential in order to check compatibility with the application since some libraries of the operating system could be necessary for the execution of certain programs.
- *Rendering engine* - to describe the engine that can interpret some source code to generate the final suitable interfaces. For our proposed model such information is considered among the most important ones as we are working within a MDE context, where the target platform has to interpret the source code generated automatically.

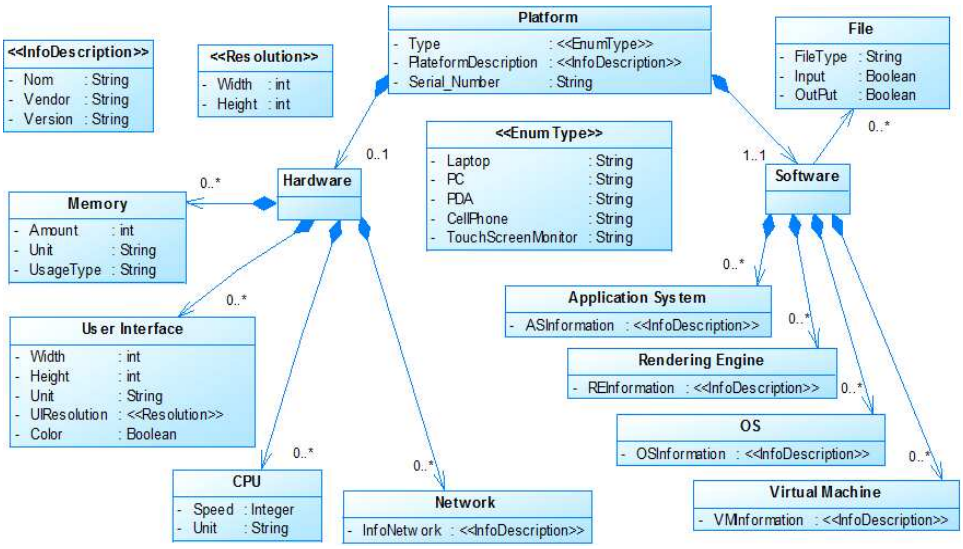

Figure 3. The platform model

3.1.3 Environment Modeling

This model (Figure 4) describes all information about the environment where the interaction takes place between the user and the platform. Most of the information related to this model are dynamic and can impact the content to be presented.

While analyzing the state of the art of this dimension (Korpipää et al., 2003; Kostadinov, 2008; Arabshianand and Schulzrinne, 2006 and Preuveneers et al., 2004), we noted that it is composed of two main classes. The first one, named *Location*, refers to the place where the user is located at the time of interaction with the platform. This place can be described in a deterministic manner through the use of *geometric* data (such as GPS coordinates, city, street,

etc.) or through *symbolic* data relative to another geometric location (opposite, next to , …) as proposed in Becker and Dürr (2005). Li et al. (2007) , affirms that among challenges that faces a location-aware application designer, is that users environment changes dynamically and it is not static. This idea was also proposed by Gu et al. (2005) indicating that to develop location-aware applications, designers have not only to model physical aspects (Persons and objects) but they have also to use this information in the definition of proactive services a to make adaptation more suitable and more intelligent.

The second one considers the *time*, which indicates the moment of interaction with the platform (Arabshianand, 2006; Korpipää et al., 2003 and Hobbs and Pan, 2006). By analogy with the location dimension, time could be described by the *exact time* (year, month, day, ...) or by a *symbolic time*; that is, a description such as summer, school holidays, etc.

Besides *Location* and *Time*, some authors (Korpipää et al., 2003; Lin et al., 2005; Preuveneers et al., 2004; Kim and Choi, 2006 and Schmidt et al., 1999) include additional information to describe the environmental dimension (such as weather, sound, etc.). This additional information related to the environment was integrated into our model as a class named *Environmental Condition*.

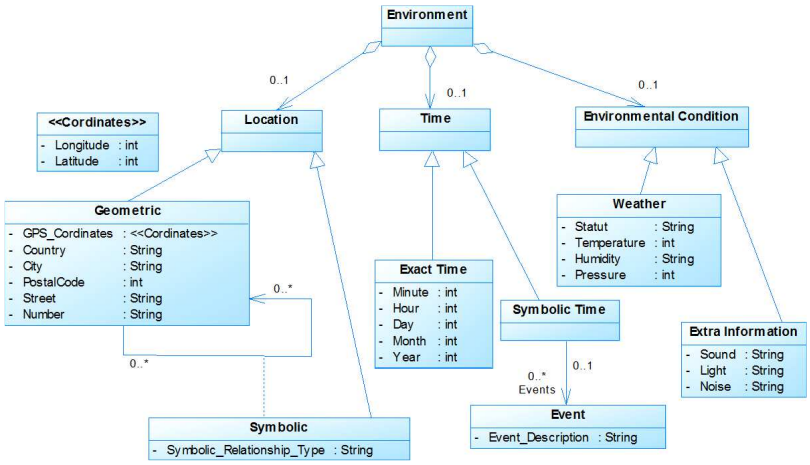

Figure 4. The environment model

3.2 Mapping Context Model with Application Domain

Once a context model is specified, the next question to address is how it could be used in a specific application domain, since personalization should be done in specific application domains. For example, with respect to Figure 1, how can we set that “city” is the city where the user is at the time of using the system or that, for the proposed itineraries, we should consider the age and the ability of the user? To address this problem we propose to map the context related concepts to the specific concepts of the application domain.

To establish the mappings we assume that the vocabulary of an application domain is defined in a domain ontology. Domain ontologies (Guarino, 1998) express conceptualizations (i.e., description of entities and their properties, relationships, and constraints) that are specific to a domain (e.g., medicine or transportation) to be used in several applications from this domain.

Considering the elements of a domain ontology (i.e., concepts, properties, relationships, and axioms) and those of a context model (i.e., concepts, attributes and relationships), we propose a meta-model (see Figure 5) that sets a mapping between any element from a context model and any element from a domain ontology, except for the constraints. The constraints express rules to infer new instances from concepts or new concept classifications, i.e., the concepts/attributes, which are used in the mappings. Such meta-model exploits three types of mappings:

- *Direct mapping*, when some information of the domain is directly associated with the information modeled in the context in that they have the same meaning. For example, any information to identify the user in the domain ontology is directly associated with the name of the user in the context. The departure city name in Figure 1(a) is another example of direct mapping. To define a direct mapping the designer just look in the domain ontology if there is any information that is the same of some information defined in the context model;
- *Indicative mapping* - when some information of the context indicates the presence or absence of some information in the domain. For example, the information about a user interest such as s-he practices cycling as sport can indicate the kind of book s-he can be interested in a domain of bookstores. The preferred transportation mode in Figure 1(a) is also an example of indicative mapping. To define an indicative mapping the designer should verify for each context element defined as a *Boolean* attribute if there is some concept in the domain that represents that context element;
- *Indirect mapping* - when some information of the domain ontology is influenced by some information in the context model. For example, the price of travelling for seniors or students is indirectly associated with demographic information about age that will influence the search for prices. The proposed itineraries in Figure 1(b) are another example of indirect mapping where the itineraries are indirectly associated with the age and abilities of the user. To define an indirect mapping the designer should verify if there is any information in the domain ontology that could vary depending on some personnel information modeled in the context.

Note that in Figure 5 any element of an ontology (concepts or attributes) can be mapped onto any element of the context of use (concepts or attributes). For each mapping, the type (direct, indicative or indirect) should be set. Note also that we can have domain concepts without any mapping, or with more than one mapping.

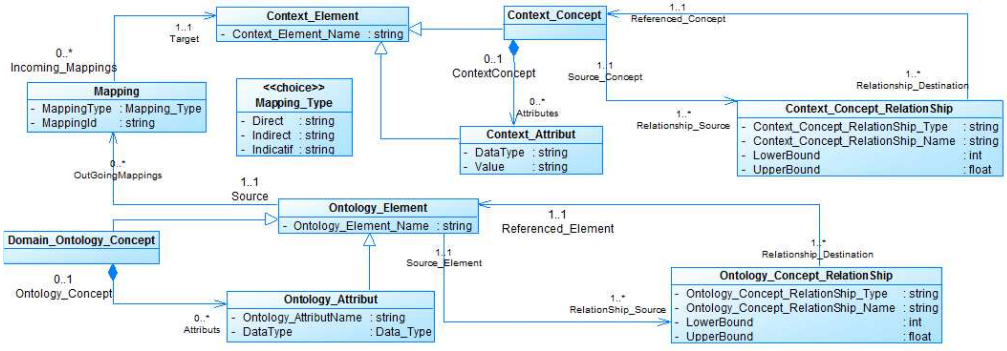

Figure 5. Meta-model for mapping context and ontology elements

4. USING CONTEXT MODELING AND ONTOLOGY IN A MDA APPROACH

In the last decade, Model-Driven Architecture (MDA) has gained attention from the human-computer interface community, because of its capability of code generation from abstract models and transformations. Successful MDA-compliant tools (e.g., OlivaNova, Teresa, UsiXML toolkit) automatically generate user interfaces personalized for specific platforms, considering design elements such as fields, screen resolution, screen size, and so on. Due to those reasons (capability of working with models from the beginning of UI conception till the code generation and the successful use of this technology for UI purposes), we decide to explore MDA to address our goal of including some content personalization from the beginning of the UI design (Bacha et al., 2011). Recall that MDA is an approach for specifying a system independently of the platform that supports it; and for transforming the specification into a software system, for a particular platform. To do so, three viewpoints of a system are specified by different models: computation independent model (CIM), that focuses on the requirements for the system; platform independent model (PIM) that specifies a degree of platform independence suitable to be used with different specific platforms; and, platform specific model (PSM) that combines the specifications in the PIM with details that specify how that system uses a particular type of platform. Transformations are used to convert a model to another model of the same system (from CIM to PIM, and from PIM to PSM).

In this section we present the models of our MDA approach (section 4.1), the models transformations (section 4.2) and an illustrative example.

4.1 Approach Models

In our approach (Figure 6), the Context model, the Domain ontology and its mapping are the core for the generation of a personalized UI considering its containers and content. Once the user connects to a system application (runtime), a specific module of the system receives his/her identification and generates his/her personalized information as an instance of our Context Model. Since the context model and the mapping with the domain are used during design, the final interface will use the instance to provide the interface with personalized contents in its input/output fields.

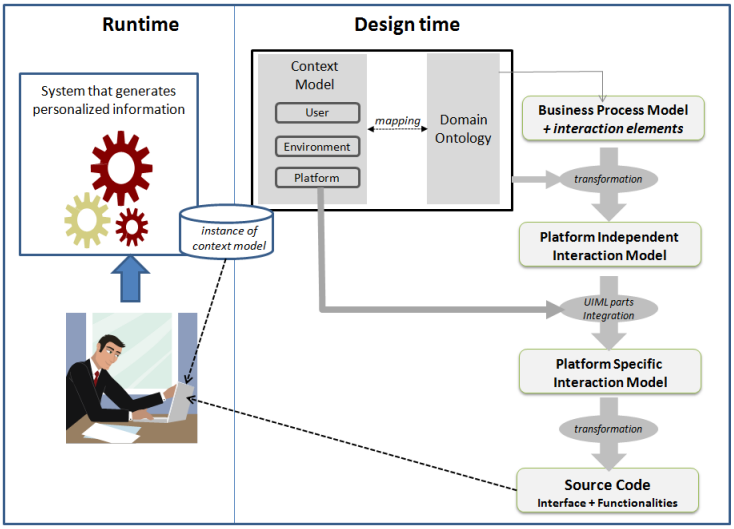

Figure 6. Our MDA approach for user interface personalization

During design time, the first main model in the MDA structure is the Business Process Model (BPM) that allows the definition of tasks to support the business goal (interactive tasks, non-interactive tasks and manual tasks) and the information flow between tasks since we are interested in content personalization. We chose to use BPMN notation (Business Process Model Notation) because of its capability to model the passage of information flow and the dynamic application aspect. The idea of using BPMN at the CIM level is also shared by other authors (e.g., Touzi et al., 2008; Rodriguez et al., 2007 and Brossard et al. 2007). As we model the tasks, they should be annotated with:

- the concept of the domain ontology, whenever possible, and its pertinent mapping with the context model. In this case, we can use a predefined mapping between the domain ontology and the context model elements, or to define a new mapping for specific purposes.
- interaction elements, in the case of interactive tasks. Interaction elements are an abstract view of type of interaction with the user such as: different type of input of information (informed by the user – named *UIFieldManual*, selected from a defined set of information – named *UIFieldOneChoice*, etc.), output information (named *UIFieldOutput*) or the idea of a group of information (named *UIUnit*).

At the PIM and PSM level, there are respectively two models: the Platform Independent Interaction Model (PIIM) and Platform Specific Interaction Model (PSIM). Those models are specified in UIML (User Interface Markup Language) (Helms et al., 2009). We chose UIML as a language for PIM and PSM levels representation because it provides the tools development for the creation of platform independent interfaces. Indeed, the conversion from UIML to code in different platforms is already provided (for example, the LiquidApps toolkit implements the conversion from UIML to Java, HTML, WML, VoiceXML).

An UIML model is composed of two main components: `interface` and `peers`. The `interface` component represent the description of the interface threw four parts: `structure`, that represents the organization and hierarchies of all UI parts; `content` that

describes the set of the application information that will be displayed (e.g. in different languages), `behavior` that represent the behavior of the application at the user interaction time, and `style` that defines all properties specific for each UI element. The `Peers` component links the generic UI elements and their properties, to a specific platform using the `presentation` part. Indeed, it describes the calling conventions for methods that are invoked by the UIML code in the `Logic` part. The `Logic` part links methods that are used in UIML with other ones used in a platform-specific source code.

The PIIM is composed of the `structure`, `behavior`, `content` and `style` parts. To manipulate the content, UIML offers two choices: either to integrate it within the `style` part, or to separate it under the `content` part. The second alternative is useful only if designers have several contents for an interaction element and only if the contents are already known. For that reason, we decide to adopt the first choice by integrating the `content` part within the `style` one. In the PIIM, the `style` part contains only properties related to content.

The PSIM is composed of the `style`, `presentation` and `logic` parts. The `style` part here contains the layout information using the appropriate `style` properties based on the chosen platform. The `presentation` part serves to map generic UIML classes with platform-specific ones and the `logic` part contains mappings between the methods used in the `behavior` part and those that will be used on the platform-specific source code.

4.2 Models Transformation

Transformations were written using ATL (ATLAS Transformation Language). During the first transformation (from BPM to PIIM), in addition to the BPM we use the mapping metamodel, the context model and the domain ontology as input. The following parts are generated in the PIIM : the `structure`, the `behavior`, and properties related to content manipulated by the `style` part.

To generate the `structure` part we defined the UIML code that should be set for each BPMN element used in the BPM and the specific interaction element. In general, for each interaction element, a UIML `<class>` is created under the `<part>` clause. For example, for each "*Pool*" element in the Business Process Model, a `<part>` element is created with the `<class>` attribute `G:TopContainer`. Then, we analyze all elements that compose this "*Pool*". For each element, a `<part>` is created with the correspondent class attribute, and with the same "*id*" from the BPM element.

The `behavior` part is created for each BPM tasks that are annotated with the associated domain ontology concepts and the corresponding mapping type with the context. To set the behavior, we used the UIML `rule` statements which are composed of a set of `condition`s and associated `actions`. The `condition` is used to keep the dynamics of the application modeled in the BPM when transforming to UIML. This is done by the use of activation variables that controls when the task (or other elements) will be performed and after that which next elements should be activated to be, then, executed. An `action` (i.e., `when-true` statement) is defined for each kind of mapping as follows:

- for direct mappings the value of the context concept mapped to the used domain concept is set to the input/output field depending on the kind of interaction element. For instance, for the information informed by the user (i.e. *UIFieldInManual*) and that the mapping

type is "*Direct*", the `when-true` part sets that the "`g:text`" property will be filled-in automatically by the value taken from the context by calling the `GetValueFromContext` method.

- for the indicative mapping, the value of the context element (true/false) is verified to decide the value (selected/not selected) of the interaction element. This is done by including a `condition` statement in the UIML `rule` that verifies the value of the context element mapped to the ontology element. The generated `when-true` statement, set the `g:selected` property of the created UIML to true.
- for indirect mappings, a <`call`> UIML statement is generated in PIIM under the `when-true` statement. The <`call`> statement represents a call to an external method or function (that uses a language other than UIML). It defines which information should be returned based on parameters (elements indirectly associated to the domain concept). It is done by a definition of a `Get_element` method where the first parameter is the information to be searched and the other parameters are the criteria that should be taken into account.

In the `behavior` part, the generated UIML code manipulates `style` properties that are related to content (such as `g:text`, `g:selected`).

From PIIM to PSIM we do a transition and not a transformation. We named transition because we do not generate code from the information of PIIM but rather we integrate remaining UIML parts related to the target platform. The transition from PIIM to PSIM considers characteristics of specific platforms (e.g. desktop, IPhone, etc.) by integrating specific remaining `style` part layout properties for each UI element. Moreover, for each `call` generated for the indirect mappings at the PIIM, a `logic` statement will be added with the information about the implemented code for this method. This code is implemented by the software designer to search the required information based on the defined parameters.

Figure 5 shows an example of ATL rule for a task to which is associated a Direct mapping (as verified in line 19-20). We note that ATL code lines (from 23 to 26) generate `condition` statement and (from 27 to 38) generate `action` statement.

```
1.  module TransformationCode;
2.  create OUT : uiml from IN : bpmn, IN1 : mapping, IN2 : StaticUI;
3.  --- The Input and output metamodels
4.  rule UIMLRulefromTask {
5.  from
6.  bpd :bpmn!BusinessProcessDiagram
7.  using{
8.  --- Variables declaration …
9.  }
10. do {
11. …
12. for (task in pool.ContainedElements)
13. {
14.  if (task.ContainedElementType = #UserTask
15. and   task.Related_Static_Element.oclIsTypeOf(StaticUI!UIFieldInManual))
16.  {
17.   for ( mapp in mapping!Mapping.allInstancesFrom('IN1'))
18.    {
19.    if (task.RelatedOntologyElement.OntologyElementName = mapp.Source.Ontology_Element_Name
20. and mapp.MappingType = #Direct and task.RelatedOntologyElement.MappingType = #Direct)
21. --- Filling the behavior part if conditions are True
22.     rulle <- thisModule.createBehaviorRule ('OnlyActivatedrule'+ task.Id.toString(), behavior);
23.     condition <- thisModule.createRuleCondition(rulle);
24.     op <- thisModule.createConditionOperation('Equal', condition);
25.     variable <- thisModule.createOperationVariable (task.Id.toString()+ 'isactivated',op);
26.     thisModule.createOperationConstant('true',op);
27.     action <- thisModule.createRuleAction(rulle );
28.     whentrue <- thisModule.createWhenTrueAction(action );
29. --- If the conditions of the “condition” part is true, generating the “when-true” part then
30. --- g:text property will be filled-in  automatically by the value taken from the context
31. --- threw the “GetValueFromContext”  method
32.     property <- thisModule.createWhenTruePropertyCall (task.Id.toString(), 'g:text',whentrue);
33.     call <- thisModule.createPropertyCall(task.Id + 'Context', task.Id + 'GetValueFromContext',
34.     property);
35. --- The parameter of the “GetValueFromContext”  method will be the name of the
36. --- ontology element related to the task
37.     thisModule.createCallParam(mapp.Target.Context_Element_Name, call);
38.     thisModule.createWhenTrueProperty(task.Id.toString(), 'g:visible', 'true' ,whentrue);
39. } } }
```

Figure 7. ATL transformation rule – example of behavior part generation (Direct mapping)

4.3 An Example of user Interface Design

Let us suppose that the interface showed in Figure 1(a) (Section 2) is part of a system for planning a trip. Figure 8 shows the BPM related to this interface. We note that for each BPM element, we define: an id, its type (user task, when it means interaction with the user; or sub-process, when it means it will be decomposed in other elements). We have also to define the related interaction element associated to it, the domain ontology concept associated and the kind of mapping, if applicable. The idea is that the departure city and preferences of transport mode should be already filled in by the system. Although, the user can change this information, the system should provide the form with personalized content collected based on the user context.

Using a public transportation ontology previously defined in (Houda et al., 2010) we mapped the information of the Context Model and the concepts of the ontology. This ontology has the knowledge about public transportation, including itineraries, stop points, cities, transport modes used, geographic elements surrounding the stop points (libraries, bank, etc). Table 4 shows some mappings defined. For indirect mappings, we set that *DirectJourneyPattern* (a kind of itinerary where the user do not need to do any connection change) is indirectly associated with the attributes *Age* and *To walk* from *Demographic Information* and *Ability* classes respectively.

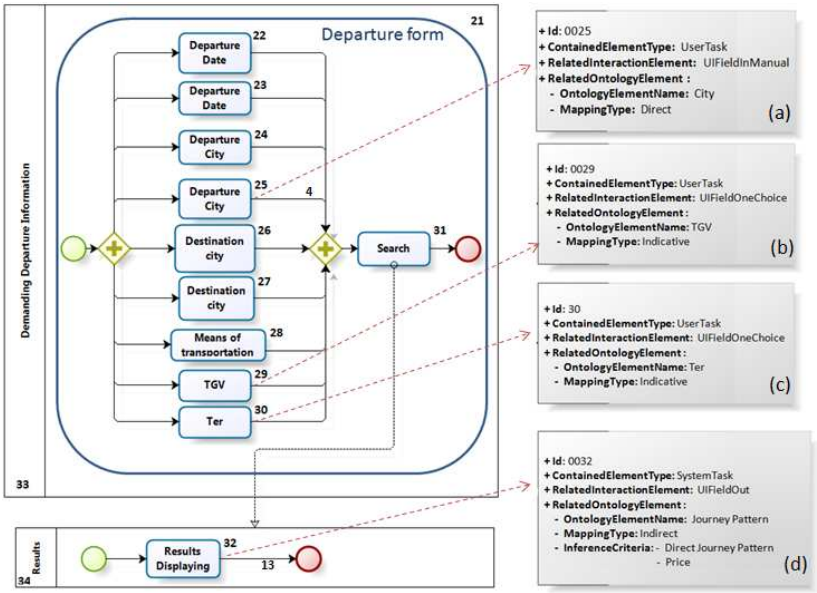

Figure 8. Business Process Model example

Table 4. Example of Domain ontology and Context of Use mapping

Domain ontology concept	**Mapping type**	**Context element**
City	Direct	City (from *Geometric*)
Transportation Mode	Indicative	TGV and TER (from *Train*)
Direct Journey Pattern	Indirect	To walk (from *Ability*)
Price	Indirect	Age (from *Demographic Information*)

In Figure 8, we identify the following input fields: departure date, departure city, destination city and the means of transportation (*TGV* - high-speed train or *TER* - Regional Express Train). We note that the task 25 named *Departure City* is associated to the ontology element *City* with a direct mapping (Figure 8(a)). The mapping was already set as it is presented in Table 4. That means that the value of the input field of the city shall be filled automatically by the value of the context element mapped directly to the concept of ontology *City.* In the same way, the modes of transportation were set as indicative mapping with the correspondent ontology concept (Figure 8(b) and (c)). It means that the selection or not of this element depends on the value of the context element mapped to the concept of ontology. In this example, we have associated to the task (0029) the *TGV* ontology concept to show that the choice of option depends on the user preferences (prefers or not travelling by TGV). Finally, the task named *Results Displaying* serves as an output of information. We associated with this task an indirect mapping where the searched concept is *Journey Pattern* (a concept ontology that represents the path of the train). Since it is an indirect mapping the criteria to define the searched concept was established with two concepts: *Direct Journey Pattern* and *Price* (Figure 8 (d)). Since the concept *Direct Journey Pattern* is associated with the context attribute

ToWalk (see Table 4), if the value of the attribute *ToWalk* = `true`, then the system must search paths of all types, else (if the user cannot walk) the criteria *ToWalk* the will be taken into account during the research process and the system should provide only direct journey patterns.

Figure 9 presents a part of the transformation result from BPM to PIIM. We note the `structure` and `behavior` PIIM parts that were generated with the transformation rule defined in ATL and presented in Figure 7. As a direct mapping, the property `g:text` of the element 25, is filled in with the value of the *City* attribute deduced from the context model through the `0025GetValueFromContext` method. After that the element 25 should transfer the activation to the next element.

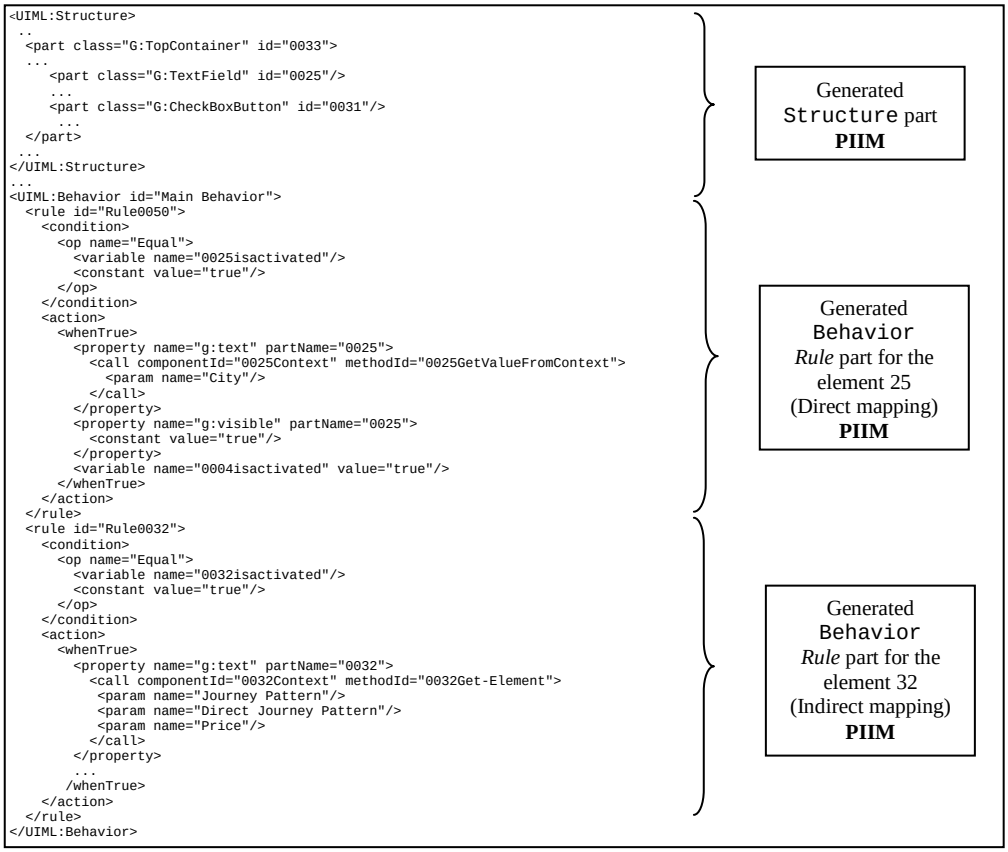

Figure 9. Example of PIIM generated by transformation from BPM to PIIM

Figure 9 presents also the PIIM generated for the indirect mapping. Since this element will serve to display information, in the generated `when-true` part, the method `0032Get-Element` is called threw the `call` statement, and it has as parameters, firstly the searched element (*Journey Pattern*) followed by the parameters that the system should consider during searching process (*Direct Journey Pattern* and *Price*).

Figure 10 shows part of the result of the transformation from PIIM to PSM supposing that the target platform will have Java as a programming language. The generated PIIM

`structure` class named `G:TextField` will be mapped to the `JTextField` Swing library class. The generated method named `0025GetValueFromContext` is mapped to the platform-specific method named `Lamih.Context.GetValueFromContext` that allows getting information from context. The `0032GetElement` method sets the method to search an element (`param1_32`) considering two criteria (`param2_32` and `param3_32`).

```
...
<UIML:Peers id="MainPeers">

 <UIML:Presentation id="MainPresentationPart">
    <d-class id="G:TextField" used-in-tag="part" maps-type="class" maps-to="javax.swing.JTextField">
     <d-property id="text" maps-type="setMethod" maps-to="setText">
       <d-param type="java.lang.String"/>
     </d-property>
    <d-class
 </UIML:Presentation>

 <UIML:logic id="MainLogicPart">
    <dComponent id="0025Context" mapsTo="Lamih.Context">
        <dMethod id= "0025GetValueFromContext" mapsTo="Lamih.Context.GetValueFromContext">
          <dParam id= "param0025GetValueFromContext" type="String"/>
        </dMethod>
    </dComponent>
     <dComponent id="0032Context" mapsTo="Lamih.Context">
        <dMethod id= "0032Get-Element" mapsTo="Lamih.Context.Get-Element">
          <dParam id= "param1_32" type="String"/>
          <dParam id= "param2_32" type="String"/>
          <dParam id= "param3_32" type="String"/>
        </dMethod>
    </dComponent>
 </UIML:logic>
</UIML:Peers>
```

Integrated `Presentation` part **PSIM**

Integrated `Logic` part **PSIM**

Figure 10. Example of code integrated to PSIM

5. RELATED WORK

Personalization or adaptation of user interfaces has been studied by different research groups in Human-Computer Interaction community. Some of the most known studies are:

i. TERESA (Berti et al., 2004) that is interested in the automatic interfaces generation problem for multi-devices applications; e.g., PDA, phone, computer. The generation is done through a set of heuristics that allow decomposing the application into a set of workspaces. Only the platform is taken into account in the design time. That means, like our approach, the generation of containers UI is based on the kind of platform.

ii. The CAMELEON framework (Calvary et al., 2003) for the generation of design time and runtime plastic applications based on a context of use. This framework is composed of four levels of models and transformations between two of those models, similar to MDA approaches. As previously defined (section 3.1), we defined the context model with the same dimensions of the context of use; that is, user, environment, and platform. However, since CAMELEON is a reference framework, no context model has been developed leaving to the designer of a specific approach to define it.

iii. UsiXML (Vanderdonckt, 2005 and Limbourg et al., 2005), an environment based on the CAMELEON framework, generates UI for different multimodal, multilanguage, and multi-context platforms. Similarly to our approach, UsiXML is MDA-compliant. However, the environment model take into account only three aspects (light - noise – stress) and the user model is defined in a meta-level that should be specified for each application domain.

iv. Bouchelliga et al., (2010), follow a more extended version of CAMELEON, to propose a

MDE approach for generating plastic interfaces for workflow information system. Unlike our purpose, it considers only the container aspect when adapting interfaces.

v. Sottet et al. (2007) define an approach for adaptive generation of plastic human-computer interface during runtime. That means, they propose to do containers adaptation at the moment the user is using the system. This idea of defining the kind of adaptation during UI design in a way that can be dynamic used during runtime is similar to our approach, since we consider that we set all mappings and generic transformation to allow the dynamic content personalization at the moment the user is using the system. However, for them this dynamic aspect is done for the UI container elements, and for us, for the UI content elements.

vi. SUPPLE tool (Gajos et al., 2010) aims to generate an interface adapted to user, his preferences and his abilities by associating the suitable widget with the appropriate user. Unlike the previous approaches, and similar to ours one, this method consider effectively the user preferences and abilities during interface adaptation. Nevertheless, it adapts only the interface presentation and does not consider the environment during the transformations.

vii. Brossard et al. (2007) propose a methodology for the development of personalized information system in the transportation domain using a MDA approach. This methodology suggests the use of fourteen models including domain ontology, user model, geographical model and a model of external systems. The personalization is defined by the inclusion of business rules in the task modeling that uses the concepts from the domain ontology. This work is similar to ours since it considers some kind of content personalization, however, all models, and in particular, the user, external systems and geographical models that could be considered as context are not designed, as his goal was only to define the methodology.

Table 5 summarizes the differences among the approaches presented above.

Table 5. Approaches Comparative table

Characteristic		**(i)**	**(ii)**	**(iii)**	**(iv)**	**(v)**	**(vi)**	**(vii)**	**Our approach**
Context model development	User			✓	✓	✓	✓		✓
	Environment			✓	✓	✓			✓
	Platform	✓		✓	✓	✓	✓		✓
Adaptation	Container	✓	✓	✓	✓	✓	✓		✓
	Content							✓	✓
When ?	Design time	✓	✓	✓	✓	✓	✓	✓	✓
	Runtime		✓	✓		✓		✓	✓
MDA-compliant				✓				✓	✓

6. SHORTCOMINGS OF THE APPROACH

As presented in the previous section, our approach provides a way for developing a full personalization (containers and content) since the design of interactive system. As an MDA-compliant approach, this facility can brings productivity for the system production since we are integrating personalization concerns during the whole development process, being reusable for several applications in the same domain. However, several limitations could be identified:

- Need of a domain ontology – Usually the MDA approaches for UI design consider a domain application model that is defined specifically for the software system to be developed. We chose to use domain ontology because of its nature of defining the domain independently of the system being developed, and therefore able to be reused for the development of various applications of the same domain. Nevertheless, we pay the effort of having an ontology.
- Need of a mapping between ontology and context model – This mapping requires a deep knowledge of the application domain in order to choose which concept should be mapped with which context element and how it should be mapped. However, once we have defined these mappings, this knowledge can be used in several applications. Therefore, this approach is justified only for domains where we wonder to develop several software systems (same reason for the choice of ontologies), or that we wonder to code the same application in several kind of platforms. If we plan to develop a single and specific software system, it will be easier to do the personalization in a different way.
- Need of addition codification for indirect mappings - For direct and indicative mappings the transformations already include directly which context information should be considered. However for indirect mappings, we just prepared the interface methods with the respective input parameters to be used in the queries that should be written in the specific source code language.
- UIML dependency – Our approach generates the code in UIML, this code must therefore to be translated to the specific language using UIML generators. To deal with the dynamic part of task models (that is the sequence of tasks) and the content personalization we used, for example, the notion of variables that is provided only in UIML 4.0, which does not have yet the code generation for any platform and language as we wished, and therefore limits our approach.
- Difficulty of identification that it paid-off – Since our choices (ontology, mappings and MDA) looked for being reused in several application developments in the same domain, the return on investment can take time as any other MDA approach.

7. CONCLUSION

With the large use of software systems to support all daily activities, the individual becomes increasingly dependent of the use of computerized services. To make these services more attractive to the users, it is important to provide systems that are personalized for each one of them. The user should feel like the application was developed for him/her, respecting his/her personal features and identifying the specific situation s-he is using the system (e.g., at home or public places, using a tablet PC or a mobile, etc.) to provide personalized information, which result in a direct gain of time.

We believe that to meet this need, it is crucial to consider the context modeling in the production of interactive systems. With this on mind, we defined a context model that could be used during the design of user interfaces to allow personalization of containers and content. Our goal was to set still in design time which information is important to generate personalized user interfaces. In this way, we defined a model that considers the user, the platform s-he will use and the possible environment s-he is while using the software system. All this information is used to perform a dynamic personalization of the user interface

contents at the moment of use. To that end the context was modeled in a generic way and should be associated with specific domain ontologies at the moment of the user interface design. Finally, these models are used within a MDA approach to generate by transformations the final user interface.

We are now working on doing mappings with other domain ontologies to confirm the generality of the context model. We are also developing a tool to support this approach.

REFERENCES

Abowd G., Dey A., Brown P., Davies N., Smith M. and Steggles P., 1999. Towards a better understanding of context and context-awareness. *In : Handheld and ubiquitous computing 99*, LNCS, Vol. 1707, Springer, Berlin, pp. 304-307

Anli A., 2006. *Méthodologie de développement des systèmes d'information personnalisés*. PhD Thésis. University of Valenciennes et du Hainaut-Cambrésis, France.

Arabshianand K. and Schulzrinne H., 2006. Distributed context-aware agent architecture for global service discovery. *In: SWUMA 2006*, Trentino, Italy.

Ayed D., Delanote D. and Berbers Y., 2007. MDD Approach for the Development of Context-Aware Applications. In Kokinov et al. (eds.), *Modeling and Using Context, CONTEXT'07*, Roskilde, Denmark. Lecture Notes in Computer Science 4635. Springer-Verlag Berlin Heidelberg, pp. 15-28..

Bacha F., Oliveira K. And Abed M., 2011. A Model Driven Architecture Approach for User Interface Generation Focused on Content Personalization. *In Proceedings of the IEEE International Conference on Research Challenges in Information Science, RCIS 2011*, Guadeloupe, France.

Becker C. and Dürr F., 2005. On location models for ubiquitous computing. *Personal and Ubiquitous Computing*, Vol. 9, No. 1, pp. 20-31.

Berti S., Mori G., Paternò F., and Santoro C., 2004. TERESA: A Transformation-Based Environment for Designing Multi-Device Interactive Applications. *In Proceedings of CHI 2004, CHI '04 extended abstracts on Human factors in Computing Systems*, ACM Press, Wien, Austria, pp.793-794.

Bouchelliga W., Mahfoudi A., Benammar L., Rebai S., Abed M.,2010. An MDE Approach for User Interface Adaptation to the Context of Use. Bernhaupt R. et al. (Ed.), *Human-Centred Software Engineering*, Third International Conference, HCSE 2010, Reykjavik, Iceland, October 2010, Proceedings, LNCS 6409, Springer, Reykjavik, Islande, pp. 62-78, octobre.

Bouzghoub M., 2004. *Action Spécifique sur la Personnalisation de l'Information – CNRS.*

Brossard A., Abed M. and Kolski C, 2007. Modélisation conceptuelle des IHM : Une approche globale s'appuyant sur les processus métier. *Ingénierie des Systèmes d'Information (ISI) - Networking and Information Systems*, Vol 12, pp. 69-108.

Brusilovsky P., Kobsa, A., and Nejdl, W. (Eds.), 2007. *The Adaptive Web: Methods and Strategies of Web Personalization*., Vol. 4321, Springer. [online] Available at: <{HYPERLINK http://www.springer.com/series/558}>.

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. and Vanderdonckt, J., 2003. A Unifying Reference Framework for Multi-Target User Interfaces, *Interacting with Computers*, Vol. 15, No. 3, pp. 289-308.

Chen H., Perich F., Finin T. and Jochi A., 2004. SOUPA: Standard ontology for ubiquitous and pervasive applications. *IEEE Computer Society*, pp. 258-267.

Cinquin L., Lalande P. A and Moreau N. *Le projet ECRM : Relation client et Internet*. Editions Eyrolle, Paris, 2002.

Dey A., 2001. Understanding and Using Context. *Journal of Personal and Ubiquitous Computing*, Vol. 5, pp. 4-7.

Dyche J., 2002. *The CRM Handbook: A Business Guide to Customer Relationship Management*. Addison-Wesley Educational Publishers, USA.

FIPA, *Device Ontology Specification*, 2001. [online] Available at: <http://www.fipa.org/specs/fipa00091/PC00091A.html>.

Gajos K., Weld D. and Wobbrock J., 2010. Automatically generating personalized user interfaces with Supple. *Artificial Intelligence*, Vol. 174, Issues 12-13, pp. 910-950.

Garía-Barrios V., Mödritscher F. and Gütl C., 2005. Personalisation versus Adaptation? A User-centred Model Approach and its Application. In K. Tochtermann, & H. Maurer: *Proceedings of the International Conference on Knowledge Management*, Graz, Austria, pp. 120-127.

Gu H., Shi Y., Xu G. and Chen Y., 2005. A Core Model Supporting Location-Aware Computing in mart Classroom. R.W.H. Lau et al. (Eds.), LNCS 3583, Springer-Verlag Berlin Heidelberg, pp. 1-13.

Guarino N., 1998. Formal Ontology in Information Systems. *Proceedings of FOIS'98*, Italy.

Hachani S., Chessa S. and Front A., 2009. Une approche générique pour l'adaptation dynamique des IHM au contexte. *Proceedings of the 21st International Conference on Association Francophone d'Interaction Homme-Machine*. Grenoble, France, pp. 89-96.

Hagen P., Manning, H. and Souza, R., 1999. *Smart Personalization*. Forrester Research, USA.

Helms, J., Schaefer, R., Luyten, K., Vermeulen, J., Abrams, M., Coyette, A. and Vanderdonckt, J., 2009. Human-Centered Engineering with the User Interface Markup Language, in Seffah, A., Vanderdonckt, J., Desmarais, M. (eds.), *Human-Centered Software Engineering*, Chapter 7, HCI Series, Springer, London, pp. 141-173.

Hirsh H., Basu C. and Davison B., 2000. Learning to Personalize. *Communications of the ACM* , Vol. 43, N° 8., pp. 102-108.

Houda, M.., Khemaja M., Oliveira K., Abed M., 2010. A public transportation ontology to support user travel planning, *In: IEEE Proceedings of Research Challenges in Information Science*, pp. 127-136.

Hobbs J. and Pan F., 2006. Time Ontology in OWL. *Ontology Engineering Patterns Task Force of the Semantic Web Best Practices and Deployment Working Group*, World Wide Web Consortium (W3C) notes. [online] Available at: <{HYPERLINK http://www.w3.org/TR/owl-time/}>

Jrad Z., Aufaure M. and Hadjouni M., 2007. Contextual User Modelling for Web Personalisation. *PAWI 2007 (Personalized Access to Web Information), Workshop of the 8th Conference on Web Information System Engineering (WISE 2007)*, Lecture Notes in Computer Science LNCS 4832, Nancy. pp. 350-361.

Kappel G., Retschitzegger W. And Schwinger W., 2000. Modeling Customizable Web Applications – A Requirement's Perspective, *Proceedings of the International Conference on Digital Libraries*, Kyoto, Japan

Kim E. and Choi J., 2006. An Ontology-Based Context Model in a Smart Home. *Computational Science and Its Applications*, pp.11-20.

Korpipää P., Mäntyjärvi J., Kela J., Keränen H., and Malm E., 2003. Managing Context Information in Mobile Devices. *IEEE Pervasive Computing*, Vol. 2, No. 3, pp. 42-51.

Kostadinov D., 2008. *Personnalisation de l'information : une approche de gestion de profils et de reformulation de requêtes.* PhD Thesis. University of Versailles Saint-Quentin –en-Yvelines.

Ledoux T., 2001. *Etat de l'art sur l'adaptabilité*. Research Report RNTL ARCAD, number Livrable D1.1

Li Y., Hong J.I. and Landay J.A.., 2007. Design Challenges and Principles for Wizard of Oz Testing of Location-Enhanced Applications. *IEEE PervasiveComputing*, Vol. 6, pp. 70–75.

Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L. and Lopez, V., 2005. UsiXML: a Language Supporting Multi-Path Development of User Interfaces, *Proc. of 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems EHCI-DSVIS'2004*. LNCS, Vol. 3425, *Springer-Verlag*, Berlin, pp. 200-220.

Lin X., Li S., Xu J. , Shi W., and Gao Q., 2005. An Efficient Context Modeling and Reasoning System in Pervasive Environment: Using Absolute and Relative Context Filtering Technology. *Advances in Web-Age Information Managemen,* Lecture Notes in Computer Science, Vol. 3739, pp. 357-367.

Mobasher B., 2000. Automatic personalization based on Web usage mining. *Communications of the ACM*, Vol. 43, pp. 142-151.

OMG, 2003. *MDA Guide Version 1.0.1*. [online] Available at: <{HYPERLINK http://www.omg.org/cgi-bin/doc?omg/03-06-01}>.

Preuveneers D., Bergh J., Wagelaar D. , Georges A., Rigole P., Clerckx T. and Berbers Y., 2004. Towards an Extensible Context Ontology for Ambient Intelligence. *Ambient Intelligence*, pp. 148-159.

Rodríguez A., Fernández-Medina E. and Piattini M., 2007. Towards CIM to PIM Transformation: From Secure Business Processes Defined in BPMN to Use-Cases. In Business Process Management, Vol. 4714, Springer Berlin / Heidelberg pp. 408-415.

Rousseau B., Browne P., Malone P. and Ofughlu M., 2004. User Profiling for Content Personalisation in Information Retrieval. *In ACM Symposium on Applied Computing*, Nicosia, Chypre.

Schilit B., Adams N., and Want R., 1994. Context-aware computing applications. *In Proceedings of the International Workshop on Mobile Computing Systems and Applications*, IEEE Computer Society, pp. 85-90.

Schmidt A., Beigl M. and Gellersen H., 1999. There is more to Context than Location. *Computers & Graphics Journal*, Elsevier, Vol 23, No.6, pp. 893-902.

Simonin J. and Carbonell N., 2006. *Interfaces adaptatives : adaptation dynamique à l'utilisateur courant*. In Saleh, I. and Regottaz, D., Interfaces numériques, Paris : Hermes Lavoisier (coll. Information, hypermédias et communication).

Sottet J.S., Ganneau V., Calvary G., Coutaz J., Demeure A., Favre J.M. and Demumieux R. , 2007. Model-Driven Adaptation for Plastic User Interfaces. *Human-Computer Interaction – INTERACT 2007 In Human-Computer Interaction –* , Vol. 4662 , pp. 397-410.

Strang T. and Linnhoff-Popien C., 2004. A Context Modeling Survey. *Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004* - The Sixth International Conference on Ubiquitous Computing, Nottingham, England.

Taconet C. and Kazi Aoul Z., 2008. Context-awareness and Model Driven Engineering: Illustration by an e-commerce application scenario. *CMMSE'08 Worshop, In ICDIM proceedings.*

Tesoriero, R. and Vanderdonckt, J., 2010. Extending UsiXML to support User-aware Interfaces, *Proc. of 3rd IFIP Conf. on Human-Centred Software Engineering HCSE 2010 (Reykjavik, October 14-15, 2010), R. Bernhaupt, P. Forbrig, J. Gulliksen and M. Lárusdóttir (eds.)*, Lecture Notes in Computer Science, Vol. 6409, Springer-Verlag, pp. 95-110.

Touzi J., Bénaben F. and Pingaud H., 2008. Prototype to Support Morphism between BPMN Collaborative Process Model and Collaborative SOA Architecture Model. *Enterprise Interoperability III*, Springer-London, pp. 145-157.

UMO, User Model Ontology, 2003. [online] Available at: <{HYPERLINK http://www.u2m.org/2003/02/UserModelOntology.daml}>

UsiXML, USer Interface eXtensible Markup Language (version 1.8), 2007. , *Université catholique de Louvain (Eds)*, Belgium.

Vanderdonckt J., 2005. A MDA-Compliant Environment for Developing User Interfaces of Information Systems. *In Proceedings of the 17th Conference on Advanced Information System Engineerings, CAiSE 2005*, Porto, Portugal, pp. 16-31.

Van Setten M., 2001. *Personalized Information Systems*. Giga CE project part of Gigaport Project, Telematica Institut, Netherlands.

Wang X., Gu T., Zhang D. and Pung H., 2004. Ontology based context modeling and reasoning using OWL. *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, pp. 18-22.

Weißenberg N., 2004. Using ontologies in personalized mobile applications, *In GIS '04: Proceedings of the 12th annual ACM international workshop on Geographic information systems, ACM Press*, pp. 2-11.

Won K., 2002. Personalization: Definition, Status, and Challenges ahead. *Journal of Object Technology*, Vol 1, pp. 29-40.

W3C, 2009, Delivery Context Ontology (DCO). [online] Available at: <{HYPERLINK http://www.w3.org/TR/2009/WD-dcontology-20090616/}>