Technique-Independent Location-aware User Interfaces

Ricardo Tesoriero[1,2], Jean Vanderdonckt[2], and José A.Gallud[3]
[1]Computing Systems Department, University of Castilla-La Mancha
Campus Universitario de Albacete, 02071, Albacete (Spain)
[2]Louvain School of Management, Université catholique de Louvain
Place des Doyens, 1, B-1348, Louvain-la-Neuve (Belgium)
Statistics, Mathematics and Computing Department, Miguel Hernandez University
[3]ALTEA Building, Benidorm s/n, 03590, Elche (Spain)
ricardo.tesoriero@uclm.es, {ricardo.tesoriero, jean.vanderdonckt}@uclouvain.be, jgallud@umh.es

ABSTRACT
Many techniques have been developed to support location awareness in user interfaces. Most of them are expressed at the code level, making them inflexible for modification of the location-awareness logic. They are very specific to a certain domain of application, making them hard to reuse or transfer. This paper introduces a model-based approach for specifying location-awareness throughout the user interface development life cycle from the task and domain level propagated until the final user interface level. This approach is supported by a software middleware that accepts a formal definition of the location-awareness logic as input and processes it in a logical way. In this manner, multiple techniques for location-awareness could be expressed and supported, with flexibility and reusability, thus obtaining technique-independent location-aware user interfaces. For this purpose, the space concept is defined at the task and domain level and then linked to an extension of the concrete user interface model, where the middleware operates. In order to exemplify the model-based approach, some techniques for location-awareness will be expressed and a case study will be explained: the Location-aware Remote Control system.

Keywords
Location-awareness, model-based approach, region definition, space model, ubiquitous computing.

General Terms
Design, Experimentation, Human Factors, Verification.

ACM Classification Keywords
D2.2 [**Software Engineering**]: Design Tools and Techniques – *Modules and interfaces; user interfaces*. D2.m [**Software Engineering**]: Miscellaneous – Rapid Prototyping; reusable software. H5.2 [**Information interfaces and presentation**]: User Interfaces. I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods.

INTRODUCTION
Heterogeneous interacting devices and surfaces are part of our surroundings in our lives. Through the popularity of mobile devices and communication technologies, the ubiquitous computing scenario envisioned by Weiser is becoming reality [36]. Consequently, a new set of interactive applications where the context of use, defined as "any information that can be used to characterize the situation of an entity" [7], allows applications to make assumptions about users' current situation and act accordingly.

Thus, location-aware applications employ users' location to interact with the system. There are many fields in which location-aware applications are useful: home automation (e.g., location-aware remote controls adapt the User Interface to the nearest device), office automation where users are assisted by the system when locating resources (e.g., the nearest printer to print a document), m-learning applications where students perform learning activities "in-situ" (e.g., recognizing masterpieces in art galleries), outdoor guiding systems where users get information according to their position (e.g., the nearest restaurant), indoor guiding systems where visitors are guided through the building using a mobile device, multiplayer games where participants may win various points depending on where they are or where they have been through, and augmented reality.

Designing and developing location-aware UIs is still challenging today: there is no global conceptual approach for characterizing what the location is, how to properly characterize a location change, and how to express location-awareness logic. This logic is often produced programmatically, thus making it inflexible to modify. Multiple techniques for location-awareness exist based on different definitions, and relying on deployment platforms that are not interoperable, thus making them very specific to a domain, hard to reuse in another domain of discourse.

In order to address these challenges, based on related work (Section 2), this paper discusses motivations and working hypotheses for a model-based approach for location-awareness (Section 3). Based on these requirements, Section 4 defines a generic space conceptual model that enables expressing multiple location-aware techniques, rules, and logics thanks to a unified specification language, along with a model-based approach. This model-based approach consists of: (i) a series of models that capture location-awareness aspects at different levels of abstraction with a User Interface Description Language (UIDL) to specify them; (ii) a step-wise method for transforming these specifications from a high level of abstraction until code, and (iii) a middleware software that runs the location-aware logic in order to support the aforementioned method.

RELATED WORK

In this section we expose the most relevant location technologies used by location-aware applications. Then, we expose different examples of applications in different domains where location-awareness is a key issue to board. Finally, we analyze the Topiary environment used to prototype location-aware applications.

Location technologies

According to the environment where the location awareness is achieved, the systems are classified into three categories: *indoor*, *outdoor*, and *hybrid* (that operate in both).

While most outdoor location systems are based on GPS technology, the technology employed in indoor environments is very diverse. For instance, some examples of indoor location systems are: ParcTab [34] uses infrared technology; RADAR [2] relies on the RF signal strength; the Ekahau commercial product (http://www. ekahau.com/) prefers the 802.11b Wi-Fi protocol; Herecast [19] exploits Wi-Fi signal from access points; BlueBot [23] combines RFID and RF technology; BIPS [1] simultaneously exploits a LAN network with Bluetooth technology; Cricket [24] combines RF and ultrasound technology; the ultrasound Active Office technique [35] uses ultrasound technology embedded in the roof to achieve a 3D location system; SpotOn tags also employs infrared technology; LuxTrace [26] uses solar cells to capture the light level and calculate user position; SmartFloor [18] senses pressure on the floor to detect user steps; the system described in [27] uses mobile phone cameras to detect user's location.

Some hybrid approaches are: Drishti [25] combines an ultrasound system for indoor environments with a GPS for outdoor environments; PAWS [9] uses an accelerometer and an RF system based on Ekahau; Place Lab [13] combines Bluetooth, GSM, and Wi-Fi technologies; "Bridging the gaps" [12] combines an INS with infrared technology for drifting correction; the GETA Sandals [17] combine dead-reckoning sensors with passive RFID technology.

Location-aware applications

In this section we describe some examples of location-aware applications with respect to their domain.

Regarding office automation, the system described in [3] presents information according to the office room the user is located. It also provides users with the ability to search for the nearest resources and attach UNIX directories to office rooms. In [34], the mobile device is used as a remote control. A system to check people in and out from a building is presented in [28], the Dynamic Ubiquitous Mobile Meeting Board (DUMMBO) detects the presence of two or more people in a meeting room and automatically gathers information for the meeting summary, such as participants, date and conversation recording. The Conference Assistant [8] is used to schedule appointments and the Stick-e notes propose an electronic equivalence to post-its [20].

Regarding indoor guiding systems, location-aware information retrieving applications are frequently exploited to guide visitors in cultural environments (such as, art galleries and museums). These applications: [16, 5, 30] employ from RFID to IRdA technologies to carry out this task.

Regarding learning activities, the emergence of mobile learning applications used to teach "in-situ" or other places is obvious. Thus, location-awareness becomes a key factor in informal science settings and nomadic learning [10, 21].

To sum up, the use of location-aware applications is not restricted to a particular area or domain. Besides, they all use different techniques and technologies to accomplish their purposes. Therefore, we need a model that should be independent of the technology to be employed. To cope with this problem, we have studied the most relevant approaches that adhere to *Model-Driven Engineering* (MDE) and a selected set of the most relevant methods that cope with the development of context-aware applications. We observed that none of them targets the problem from the UI perspective explicitly.

The reusability issue of previously developed UIs is a key concept from a software engineering perspective that is not tackled by any of the methods we analyzed. They are focused on the reuse of the *Platform Independent Models* (PIMs) to generate source code targeted for different platforms, but they do not support the reuse of higher level models, such as, reusing a previously designed task model in a new context of use that is similar to the previous one.

The Topiary environment

Among others, Topiary [14] rapidly prototypes location-enhanced applications based on design patterns. It allows designers to create a map that models the location of people, places, and things. It uses an active map to demonstrate scenarios, depicting location contexts creating storyboards that describe interaction sequences with a wizard. While this certainly fosters rapid prototyping, there is no underlying model that captures the results of this prototype that could be further used in the development life cycle.

However, some important UI aspects were not covered by this approach: the application is not conceived with models at different levels of abstraction, leading to a huge lack of reuse of information when developing location-aware UIs. *Routes* are one of the most important aspects of location-aware applications. Topiary defines routes in a non-deterministic way, where the order of the location events cannot be specified properly. The route alternative (e.g., the capability to define the same behavior for different routes) is also not supported by the tool. To sum up, the actual approaches are not capable enough to support the development of multi-model and multi-technique UIs for location-aware applications efficiently.

Besides, the lack of explicit models for representing the location-awareness logic in the methods analyzed prevents designers from reusing parts or whole of a previously de-

signed logic. Perhaps, it may not be an important issue when modeling logic for small scenarios such as, small buildings or houses; but it may result in an interesting resource when modeling complex buildings, such as town halls or big companies or hospitals, where the complexity of the application behavior regarding the location is crucial to the application success.

MOTIVATIONS

As the result of the related work analysis we have arrived to the following motivations:

The need for a conceptual model

A location-aware UI could be considered basically as an Event-Condition-(re-)Action (ECA) system where rules express a location-aware logic decomposed into *events* (what occurred that requires some consideration of the location), *conditions* (under which circumstances do we need location-awareness, it is not necessary to do something for all changes of location), and *reactions* (what do we need to do in order to react to a change of location). Based on this decomposition, we have specialized Norman's mental model in a variation for addressing location-awareness with seven stages, each one being specialized for each entity (Figure 1):

1. *Goals for location-awareness*: any entity (i.e., user, UI, or external third party) may be responsible for specifying and maintaining a formal expression of goals for ensuring location-awareness. Although this process is ultimately intended to be beneficial for the end user, it could be achieved with respect to any location aspect. Such aspects could fall into three categories: *location-central information* (information that describes the location, such as coordinates, temperature, pressure, humidity, closeness), *location-peripheral* information (information related to the location, but not describing the location, such as vicinity, surroundings, connections with other regions), and *meta-location information* (information about location information, such as how the location is characterized). The goals are said to be *self-expressed*, *UI-expressed*, *locally* or *remotely*, depending on their locus of control: in the user's head, in the local UI, or in a remote system. In this work, we mainly assume that these goals will be expressed in the UI, with a specification language to be defined.

2. *Initiative for reaction*: this stage is refined into formulation for a reaction request, detection of a reaction need, and notification for a reaction request, depending on who is in charge: the user, the UI, or a third party.

3. *Specification of reaction*: this stage is further refined in specification by demonstration, by computation, or by definition, depending on their origin: respectively, the user, the UI, or a third-party. When the user wants to trigger a reaction to a location change, she should specify the actions required to make this reaction, such as with programming by demonstration or by designating the operations required for this purpose. When the UI is responsible for this stage, it should compute one or several reaction proposals depending on location information available (of any type). When a third party specifies the reaction, a definition of the operations required to execute the reaction could be provided.

4. *Application of reaction*: this stage specifies which entity will apply the reaction specified in the previous stage. Since this adaptation is always applied on the UI, it should provide some mechanism to support it. If the user applies the reaction, UI mechanisms should be offered such as through customization or personalization.

5. *Transition with reaction*: this stage specifies which entity will ensure a smooth transition between the UI before and after the location change. For instance, if the system is responsible for this stage, it should provide some visualization techniques for explaining the transition.

6. *Interpretation of reaction*: this stage specifies which entity will produce meaningful information in order to facilitate the understanding of the reaction of other entities. When the UI performs some reaction without any explanation, the end user may be confronted to a problem of understanding what has been changed depending on the location information type. When the user performs some reaction, she should teach the system how to interpret this reaction for future usage.

7. *Evaluation of a reaction*: this stage specifies the entity responsible for evaluating the quality of the location-awareness performed so that it will be possible to check whether the goals initially specified in Stage 1 are satisfied. If the UI maintains some specification of goals, it should be able to update these specifications according to the reactions executed. If the goals are in the users' mind, they could be also evaluated with respect to what has been conducted in the previous stages.

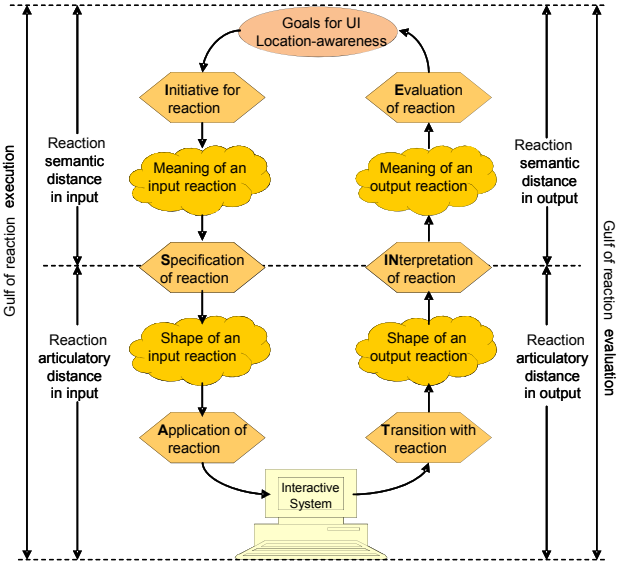

Fig. 1. Norman seven stages revisited for location-awareness.

The need for controlling the reaction

Another shortcoming found for location-aware UIs is that they can undertake reactions proactively. In terms of the model of Figure 1, it means that the application of the reactions, perhaps including the initiative and the specification that lead to this reaction, escaped from the user's control. For instance, Mobi-Timar [29] users are not allowed to change the reaction as it is pre-defined according to the required tasks for the user class depending on their location, but they can change some individual preferences, like interface layout or level of details. For instance, when a museum UI guides the visitor, it automatically prompts the description of the artwork that is closest to the visitor, thus preventing her from viewing the same artwork from a distant location. Therefore, a requirement states that the end user should be in control of the location-awareness process.

The need for gathering feedback for reaction

In the museum example, if a user comes close to an artwork and the UI does not prompt anything new, the user may become puzzled by the UI reaction. The conditions for triggering the reaction could be satisfied, but they are not observable (perhaps browsable). Or they are not satisfied and the reasons why are not observable neither. Therefore, a need arises for expressing feedback, positive or negative, but also in terms of contents (e.g., why and why not?).

The need for sociologic zone definition

Perceptual psychology may inform us to what extent a person considers an object far or close. Sociology interprets this as four spatial zones depending on the distance from subject (Figure 2). Central zone, respectively personal, social, public correspond to a focus distance from the user of 0 to 45cm, respectively 46cm to 1.2m, 1.3 to 3.6m, bigger than 3.6m [31].

The need for expressing location awareness logically

In order to express the location-awareness in a logical way, there is a need to rely on some models in order to state the ECA system. For instance, the reaction should be expressed logically on the UI independently of any location-awareness technique. For this purpose, a User Interface Description Language (UIDL) should be selected that is compatible with the Cameleon Reference Framework (CRF) [4] (a simplified version is reproduced in Figure 3, which shows the development process divided into four development steps). These steps are:

- In the *Tasks & Concepts* (T&C) step, we describe users' tasks to be carried out, and the domain-oriented concepts required to perform these tasks.
- In the *Abstract UI* (AUI) step, we define the abstract containers and the individual components [15] that will represent the artifacts on the UI. *Containers* are used to group subtasks according to various criteria (e.g., task model structural patterns, cognitive load analysis, and the identification of semantic relationships). *Individual component* represents an artifact that describes the behavior of a UI component in a modal-independent way (navigation, task performance, etc.). Thus, an AUI abstracts a CUI with respect to interaction modality.
- In the *Concrete UI* (CUI) step, we concretize an abstract UI for a given context of use into Concrete Interaction Objects (CIOs) [32] defining the widget layouts and the interface navigation. It abstracts a FUI into a UI definition that is independent of any computing platform. The CUI can also be considered as a reification of an AUI at the upper level and an abstraction of the FUI with respect to the platform.
- In the *Final UI* (FUI) step, we represent the operational UI running on a particular computing platform either by interpretation (e.g., through a Web browser) or by execution (e.g., after code compilation).

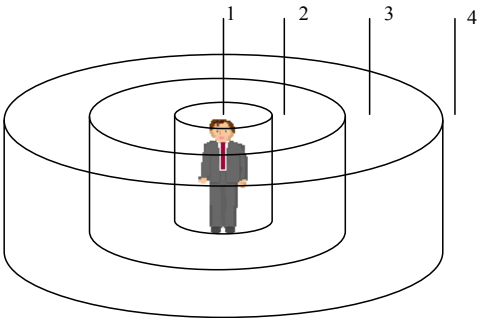

Figure 2. The four sociologic zones.

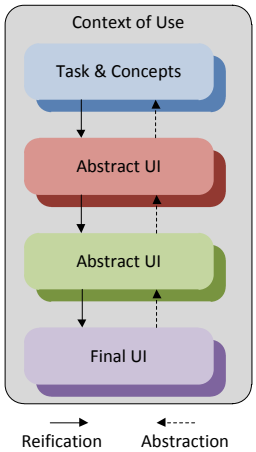

Figure 3. The simplified Cameleon Reference Framework.

UIDL candidates include that follow the CRF are: MariaXML (http://giove.isti.cnr.it/tools/Mariae/), UIML (http://www.uiml.org), UsiXML (http://www.usixml.org), or XIML (http://www.ximl.org).

We chose UsiXML [33] for the following additional criteria, but other UIDLs could be selected equally: the UsiXML modeling language must be described in terms of the Meta-Object Facility (MOF) language to enable the metamodels to be understood in a standard manner, which is a precondition for any activity to perform model-to-model (M2M) transformation and model-to-code (M2C) generation, a transformation model is also compliant with a metamodel expressed in the same way.

To support the conceptual modeling of UIs and to describe UIs at various levels of abstractions, the following set of models is defined in UsiXML [33]:

- The *taskModel* describes the task as viewed by the end user interacting with the system. It represents decomposition and temporal relationships among tasks.
- The *domainModel* describes the classes of objects manipulated by a user while interacting with a system. Typically, it could be a UML class diagram or an entity-relationship-attribute model.
- The *mappingModel* contains a series of related mappings between models or elements of models. It gathers inter-model relationships that are semantically related (reification, abstraction and translation)
- The *contextModel* describes the three aspects of a context of use in which an end user is carrying out an interactive task with a specific computing platform in a given surrounding environment.
- The *auiModel* describes the UI at the abstract level as previously defined.
- The *cuiModel* describes the UI at the concrete level as previously defined.
- The *contextModel* describes three aspects of the implementation: the user, the environment and the platform of the application. Thus, the context model wraps the other models to specify the deployment characteristics of the application.

As other UIDLs, UsiXML does not support location-awareness, but hold the appropriate concepts to express location-awareness logically. Thus, we propose to add a Space model to describe the location environment where users interact with the application. Besides, we extend the task, the AUI, and the CUI models accordingly.

THE SPACE MODEL

The space model enables the definition of the space environment the application is aware of and how the user location affects the application.

The extension affects the Task & Concepts, the CUI, the Mapping and the FUI models of UsiXML. The AUI model was not extended because the mapping between AUIs and CUIs propagates all the vital information.

While in the Tasks & Concepts layer we describe the space environment and how it affects the application, in the CUI and FUI layer we describe how the information is perceived from the environment and how it is propagated to the application.

The Task & Concept extension

The space metamodel extension for the Tasks & Concepts layer is depicted in Figure 4.

The *SpaceModel* represents the space environment the application is aware of. It is the root of the space model and it is characterized by spaces and a set of space relationships among them.

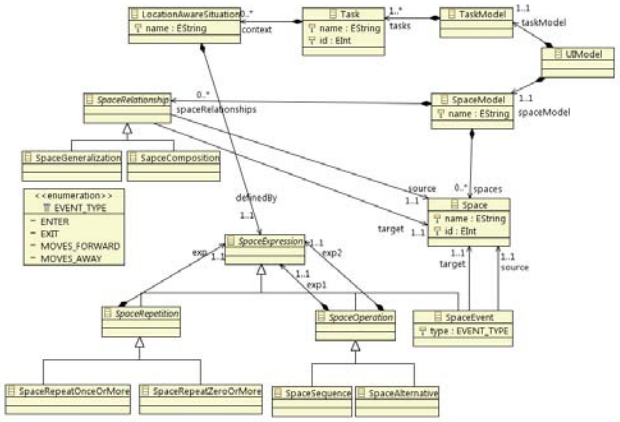

Figure 4. Space metamodel extension for Tasks & Concepts.

The space

Spaces represent location references that could be located and dimensioned by the system. The space concept covers a broad range of spaces according to the system capability. For instance, if the system is able to retrieve GPS coordinates of users and locate them in a city, a square, a building, a floor or a room, then all these physical spaces may be represented by spaces. However, the space concept is not restricted to physical space representations only. It may also be related to virtual representations, such as forms, dialogs, widgets, and so on in traditional UIs. They may also represent the virtual representation of a physical device, such as the mouse cursor. Even more, we can assign user spaces, to model colocation, or different sociological zones for the same user, as discussed in the previous section (Figure 2).

This high-level abstraction provides designers with the ability to describe "mixed" relationships among virtual and physical spaces providing highly dynamic behavior to the application. For instance: click "lights on" button in my room turn the lights on.

To provide richer location aware situations *Spaces* may be related to other *Spaces* through *SpaceRelationships*. There are two types of space relationships: *SpaceCompositions* and *SpaceGeneralizations*.

The space composition

SpaceCompositions represent a set of spaces that are treated as "the same" space. It means that if space ***S*** is composed by space ***A*** and space ***B***. If user **U** enters ***A***, then goes to ***B*** and then exits ***B***. From ***S*** point of view, the user has entered ***S*** and then exited ***S***. For instance, suppose that we have to model the personal presence in an office building. The "building" space should be composed by the "office" spaces, the "hall" spaces, the "elevator" spaces, and so on. Thus, to express that someone is outside we have to write "U out Building".

The space generalization

SpaceGeneralizations are used to express common behavior among a set of spaces. It means that if space ***S*** is gener-

alizes space ***A*** and space ***B***. If user **U** enters ***A***, then goes to ***B*** and then exits ***B***. From ***S*** point of view, the user has entered ***A***, exited ***A***, entered ***B*** and exited ***B***. For instance, suppose that we have to model who is in the rest room in an office building. As there are several rest rooms in the building, we should define the "RestRoom" space as the generalization of all the rest room spaces of the building. Thus, to express that someone is in the rest room we have to write "U in RestRoom".

The location-aware situation

The *LocationAwareSituation* is a key concept in the space model because it is "the link" between location awareness and the TaskModel. Through this relationship, and the relationship between the TaskModel and the DomainModel, the location information reaches to the core of the system. Thus, while the *Space* and the *SpaceRelationship* describe the space environment, *LocationAwareSituations* define how the environment affects system and user tasks. These situations are defined by *SpaceExpressions* that describe the situation.

The space expressions

Space expressions describe a location-aware state of the system in terms of regular expressions that are modeled following the Composite Design Pattern [11].

The space event

The simplest *SpaceExpression* is the *SpaceEvent*. It represents an event that relates two *Space*s. Note that one of them is usually the user. There are 4 types of *SpaceEvents*, according to the patterns exposed in [14]. For instance: if **U** represents the *Space* for a user, and **X** an arbitrary *Space* (e.g., a Room), the *SpaceEvent* expression ***U ENTER X*** occurs when U entered in X. Or if **U** represents the *Space* for a user, and **X** an arbitrary *Space* (e.g., a Room), the *SpaceEvent* expression ***U EXIT X*** occurs when U left X. Besides, if **U** represents the *Space* for a user, and **X** an arbitrary *Space* (e.g., a sculpture), the *SpaceEvent* expression ***U MOVES_FORWARD X*** occurs when U is approaching to X. Finally, if **U** represents the *Space* for a user, and **X** an arbitrary *Space* (e.g., a sculpture), the *SpaceEvent* expression ***U MOVES_AWAY X*** occurs when U is going away from X. Note that exiting a region does not necessarily imply an entering in the next region: it depends how regions are configured together.

The space operations

*SpaceEvent*s can be composed into *SpaceOperation*s. A *SpaceOperation* represent a relationship between two *SpaceExpression*s (*exp1* and *exp2*). There are two types of *SpaceOperation*s: *SpaceSequence* and *SpaceAlternative*. While the *SpaceSequence* represents a "path" of *SpaceEvent*s, the *SpaceAlternative* represent a "selection" between two events.

The space sequence

The space sequence is used to identify patterns of behavior. The tracking of the user position may result in a very valuable resource for both, controlling and gathering feedback for reactions. *SpaceSequences* can be used to make predictions and subsequently, ask the user for a decision (controlling reaction) or perform an action (provide feedback to the user). Thus, let ***Seq (E_1, E_2)*** be the *SpaceSequence* expression where E_1 and E_2 represent expressions, too. The situation ***Seq(U ENTER S,U ENTER S)*** occurs if the user has entered and then exited space ***S***. For instance, we want to be aware if the user **U** has gone to the cinema, to suggest him/her a place for dinner. ***Seq(U ENTER CINEMA, U EXIT CINEMA).***

The space alternative

The space alternative is a valuable resource when dealing with the same *LocationAwareSituation* in different *Space*s. Thus, let ***Alt (E_1, E_2)*** be the *SpaceAlternative* expression where E_1 and E_2 represent expressions too. The situation ***Alt(U ENTER S_1, U ENTER S_2)*** occurs if the user has entered space S_1 or space S_2. For instance, "set light intensity" to user defined if the user enters into the kitchen or the dining room ***Alt(U ENTER KITCHEN, U ENTER DINING)***.

The space repetitions

The *SpaceRepetitions* expressions are analogous to the Kleene Closure in regular expressions. They are used to express repetition of expressions. There are two types of space expressions: *SpaceRepeatZeroOrMore* and *SpaceRepeartOneOrMore*.On the one hand, the *SpaceRepeatZeroOrMore* represents an optional repetition of a *SpaceExpression*; it means that the expression may occur or not, if it does it may occur more than once. On the other hand, the *SpaceRepeatOneOrMore* represents a mandatory repetition of a *SpaceExpression*; the expression occurs at least once. Therefore, let ***Rep(E)*** be the *SpaceRepeatZeroOrMore* expression where ***E*** expression too. The situation ***Rep(Seq(U ENTER S, U EXIT S))*** occurs if the user has entered and exited ***S*** more than once or has not entered and exited ***S*** at all. Similarly, let ***Rep1(E)*** be the *SpaceRepeatOneOrMore* expression where ***E*** expression too. The situation ***Rep1(Seq(U ENTER S, U EXIT S))*** occurs if the user has entered and exited ***S*** more than once. Finally, we present a shortcut to a very common *SpaceExpression*: **U *IN S* => *Seq (U ENTER S, U EXIT S).*** For instance, suppose that we want to suggest a visitor V, in an art gallery, pieces of the same author. If all pieces of author A are generalized by the **A** *Space*, and the visitor space is represented by **V**, then the expression **Rep1(V ENTER A)** represents that the user have visited at least one piece of A.

The mapping, CUI and FUI extensions

These extensions deal with the lower level representation of the location awareness and the relationships among high-level and low-level models of the system. The Figure 5 shows the extension. The information is perceived by the system through *Sensors*. These *Sensor*s provide data that is interpreted by *LocationInterpreters*. These *Interpreter*s are

in charge of turning the sensor data into space information that can be processed by the location system in a platform independent way. Thus, this information is matched to *LocationAwareSituations* that are relevant to the system in order to propagate this information through the tasks to the domain model.

The space representation

The *SpaceRepresentation* provides an abstraction of the technology used by the location system. It provides a common identification for spaces in the system. Thus, when interpreters get data from sensors, they turn them into Spaces through the use of the *SpaceRepresentation* abstraction. Concretely, we defined three types of *SpaceRepresentation*s according to the type of technology being used. The *IDSpaceRepresentation* is used by technologies that base object location on IDs. The *UTMSpaceRepresentation* is used by GPS based technologies. The *SignalSpaceRepresentation* is used by RF based technologies.

The location sensor

The *LocationSensor* is seen as a Concrete Interaction Object (CIO) that is defined in the Concrete User Interface (CUI) model. It is used to represent the technology, or set of them, to be used by the location system.

As the CUI model is part of the Platform Specific Model (PSM), the inclusion of new technologies or brands should be reflected into this model. We have initially introduced the Wi-Fi, the Bluetooth, the Passive and active RFID, the ZigBee and the GPS technologies. While whole new technologies should be added as new *Sensors*, new brands of products may be sub-classified from existing technologies to improve the reusability.

The location interpreter

Location interpreters are the glue that relates the spaces defined in the space model to spaces in the real or virtual world through the location sensors. In this way, *LocationAwareSituations* perceives the space environment that is forwarded to the domain model according to their relevance. To carry out this task a *LocationInterpreter* is defined for *Sensor* technology to translate information received from sensors, and turn it into events that will be processed by *LocationAwareSituation*s to check for the occurrence of them. Regarding infrastructure of the location system, it can be either centralized or distributed because the application interface between the *SpaceRepresentation* and the *LocationInterpreter* can be defined using the Proxy design pattern [11].

CASE STUDY

This section explains how to model location-aware characteristics of the Location-Aware Remote Control (LARC) application using the extensions of UsiXML to support location-awareness.

LARC is a home automation application that allows users to control different home devices through different UIs (one for each device) that are automatically displayed when the user is at the range of the device control. The Figure 6 depicts the scenario with three devices: the TV, the air condition system and the electric blinds. The description is focused on the UI modeling because infrastructure issues are out of the scope of this article.

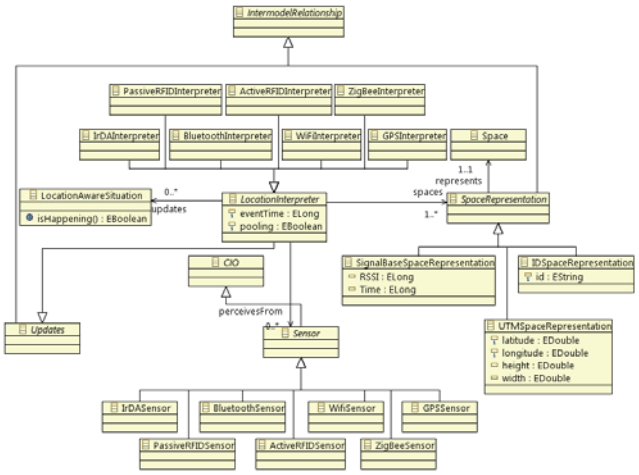

Figure 5. Space metamodel extension for CUI.

Figure 6. The Location-Aware Remote Control (LARC).

Space model

The space model is described with a UML class diagram like notation (see Figure 7). While the Space relationship is represented as an UML class without attributes or methods, the *SpaceComposition* and *SpaceGeneralization* relationships are described using the UML class composition and generalization respectively.

The physical spaces from where devices are controlled are represented by the **AC**, **TV** and **Blinds** *Space*s. They are all part of the **Room** *Space*.

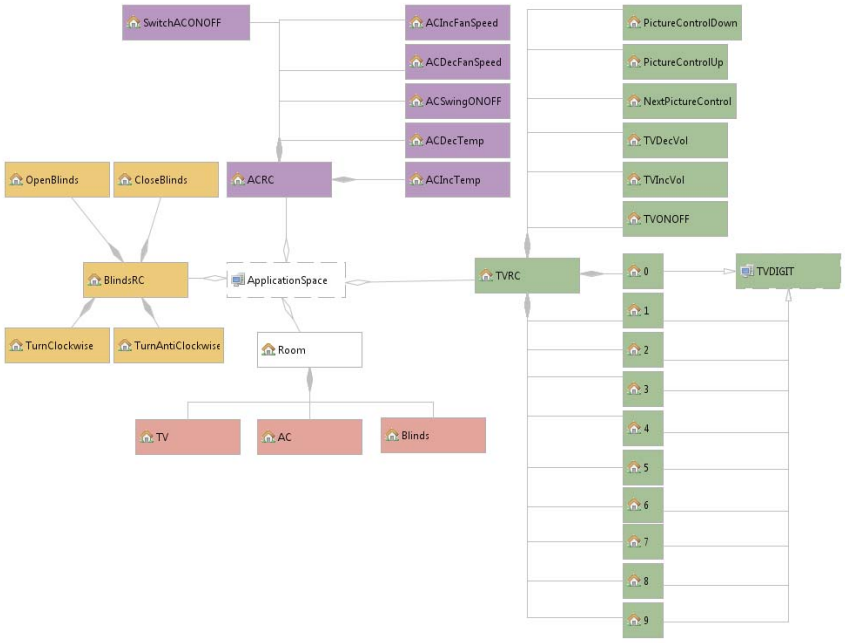

Figure 7. **The LARC space model.**

Besides, we have defined three main virtual spaces representing the UI for each device. They are the **ACRC** for the AC remote control, the **TVRC** for the TV device and the **BlindRC** for the electric blinds.The space representation for each control is very similar. Therefore, we focus our attention on the **TVRC** representation. The **TVRC** is composed by a set of spaces that may represent the controls on the PDA screen (we say "may" because we do not know how they will be represented at this stage of the development process). There spaces are **0-9**, **TVONOFF**, **TVIncVol, TVDecVol**, **NextPictureControl**, **PictureUp** and **PictureDown**. In order to select a channel, the user introduces a sequence of digits. To define this sequence we need a common identified for all digits (0-9). Therefore, we have defined the **TVDIGIT** space as the generalization of all digits.

Task model

As we have mentioned before, the description is not complete. The Figure 8 depicts the Task model for the LARC.

As UsiXML describes task models using the CTT [22] notation, we use an extension to CTT notation in order to introduce the *LocationAwareSituation concept. LocationAwareSituations* are defined within Tasks. They define the *SpaceExpressions* using the textual notation we have described on previous section.

For the sake of clarity, the Figure 8 does not show the whole task model. Although it depicts the task model for all models, the only one that is complete is the AC remote control UI. However, the complete definition for the rest of the models is analogous to the AC remote control UI.

Analyzing the model we notice that the *ShowACUI* task is executed when the *OnEnterAC* situation matches the *U ENTER AC* space expression. It means that the AC Remote Control UI will be shown when the user U enters into the AC space. This is a physical space relationship between the user and the control zone of the AC device. However, as we have mentioned before, we can define virtual relationships among physical and virtual spaces. For instance, The *SwitchACONOFF* task defines the *OnACSwitch* situation where the expression *U ENTER ACONOFF, U EXIT ACONOFF,* defines a relationship between the space that may be represented by the user's finger on the PDA screen, and may be a Button control that is being displayed on the PDA screen. Thus, the space expression defines the "on click" event on the screen.

Concrete and Final User Interface

The concrete user interface regarding location awareness is based on the *LocationSensor* concrete interaction object. The *LocationSensor* provides the user interface with the information to act. It is related to the technology (RFID, GPS, etc.) employed. This information is interpreted by the *LocationInterpreter* according to the location technique (fingerprint, triangulation, etc.) employed. Thus, using *SpaceRepresentations* that are compatible with the technology and technique employed, the *LocationAwareSituation* is evaluated according to the space expression in order to provide the user interface with the situation occurred. The implementation of the approach is based on the mappings described by the *LocationInterpreter* (*LocationAwareSituation*, *Space*) through the *SpaceRepresentation,* and the *LocationAwareSituation* mapping between (*Space*, *Task*).

IMPLEMENTATION

Both, the Task and Space model editors were developed as Eclipse plugins, using the Eclipse Modeling Framework (EMF) and the Eclipse Graphical Modeling Project (GMF). See http://www.eclipse.org/modeling/emf/ and http://www.eclipse.org/modeling/gmp/ . Both editors are built on the base of Essential Meta-Object Facility (EMOF) metamodels (http://www.omg.org/mof/), using the ECORE format. Model validation is provided by the means of constraints defined in Object Constraint Language (OCL) http://www.omg.org/spec/OCL/2.2/. Models are manipulated and stored in XMI (XMLMetadata Interchange format) (http://www.omg.org/spec/XMI/).

To build the task model editor (see Figure. 9), we have defined a modified version of the CTT where tasks can be related to different situations according to different location-aware situations. It provides designers with the ability to define space expressions for each situation. To build the space model editor (Figure 10), we have employed an UML-like Domain Specific Language (DSL) based on class diagrams.

CONCLUSION AND FUTURE WORK

This work presents how to model and develop location-aware UIs employing the UsiXML methodology. To accomplish this goal we have extended the metamodels defined by the CRF process. As discussed in the Motivation section, actual approaches to develop location-aware applications lack of technology independence, conceptual space modeling, sociologic space support, and feedback for reaction, and control for reaction, too. To cope with technology independence we have adopted a MDA approach where the PIM allows designers to define applications that are platform independent. Thus, the PSM defined by the CUI model (*Sensors, SpaceRepresentations*, etc.) abstracts the technological issues regarding the location awareness.

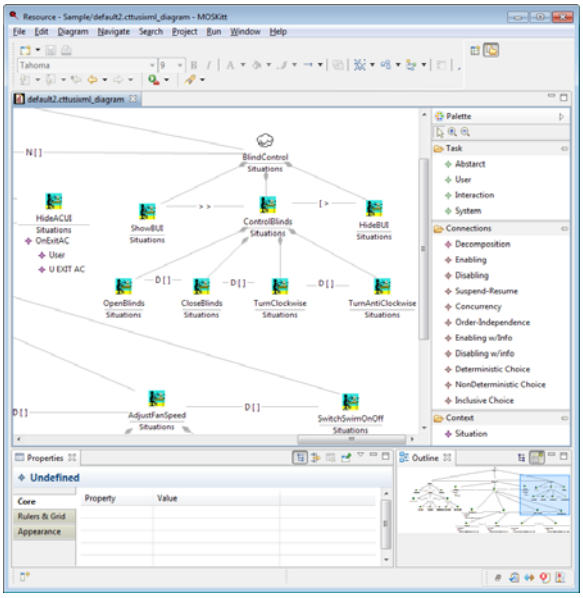

Figure 9. **Task model editor.**

The adoption of the MDA approach has also provided us with a CIM to represent the space environment using the space model where spaces can be modeled independently in a conceptual way. Regarding the need for conceptual model for spaces, the need for gathering feedback for reaction and the need for controlling the reaction, we have revisiterdingd the Norman seven stages to the space and task metamodels for location awareness as follows: the Goals for location-aware UI are defined by the Goals of the system, the *initiative for reaction* is represented by the *SpaceExpression*, the *specification of the reaction* is represented by a *LocationAwareSituation,* the *application of the reaction* is perfomed by the *Task*, the *transition with reaction* is represented by the relationship (*Task* to *Domain Model*), the *interpretation of the reaction* is exposed by the UI through the domain model, finally the *evaluation of the reaction* is performed by the User. Regarding the need for *sociological space*, we introduced a general definition of space where users have their own space that may be dynamically changed (the same user can be represented by different spaces according to the situation).The idea of the homogeneous view of spaces is also very useful to represent virtual spaces such as UI artifacts. Thus, GUIs and physical spaces are abstracted to such level that space situations may reference any of them indifferently. As future work, we are following different research lines. The position is the initial point of the gesture. Thus, we plan to introduce gestures to UsiXML methodology in order to extend its power of expression. Besides, the location awareness is just a subset of context awareness. Therefore, we are doing research on how to introduce other aspects of context awareness into UsiXML such as the physical environment or the infrastructure. Finally, the social environment is an important part of the context awareness. Therefore, the definition of the social structure of the application is a current key issue.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of the ITEA2 Call 3 UsiXML project under reference 20080026 by Région Wallonne DGO6 and the FP7-ICT5-258030 Serenoa project funded by the European Commission.

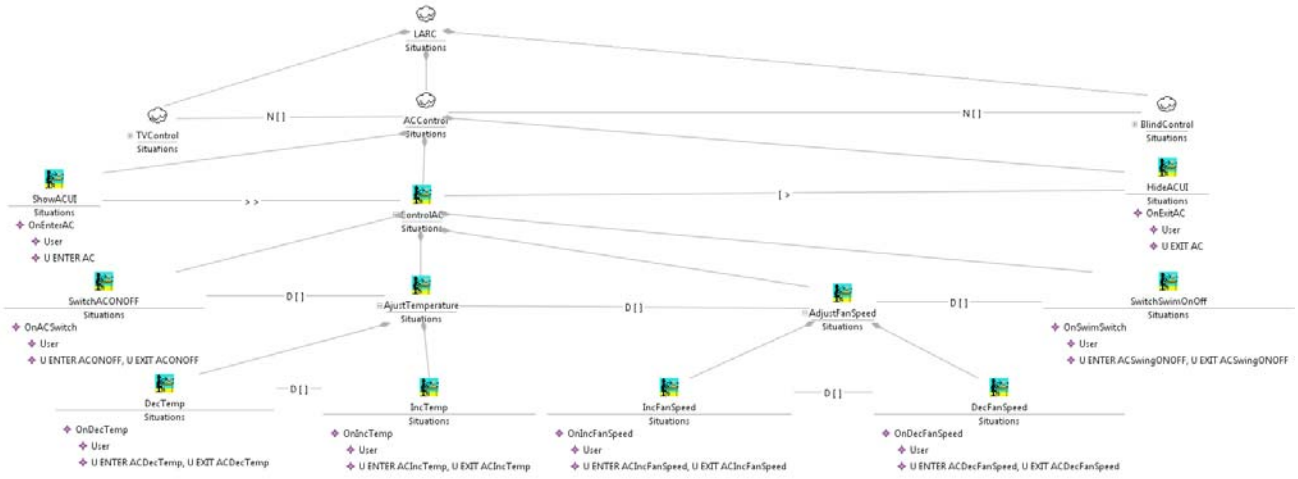

Figure 8. The LARC task model.

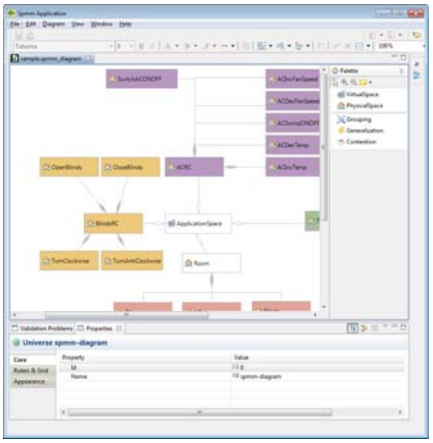

Figure 10. Space model editor.

REFERENCES

1. Anastasi, G., Bandelloni, R., Conti, M., Delmastro, F., Gregori, E., and Mainetto. G. Experimenting an indoor bluetooth-based positioning service, in *Proc. of ICDCS'2003*, 480–483.
2. Bahl, P., and Padmanabhan, V. N. RADAR: an in-building rf-based user location and tracking system in *Proc. of INFOCOM 2000*, IEEE, 775–784, 2000.
3. Brown, P. J., Bovey, J. D., and Chen, X. Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications*, 4(5), 58–64, Oct. 1997.
4. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3):289–308, 2003.
5. Cheverst, K., Davies, N., Mitchell, K., Friday, A., and Efstratiou, C. Developing a context-aware electronic tourist guide: Some issues and experiences. In *Proc. of CHI'2000*. 17–24.
6. Chou, L.-D., Lee, C.-C., Lee, M.-Y., and Chang, C.-Y. A tour guide system for mobile learning in museums. In *Proc. of WMTE 2004*, IEEE Computer Society, 195–196, 2004.
7. Dey, A. K. Understanding and using context. *Personal and Ubiquitous Computing*, 5, 4–7, 2001.
8. Dey, A. K., Salber, D., Abowd, G. D., and Futakawa. M. The conference assistant: Combining context-awareness with wearable computing. In *Proc. of ISWC*, 21–28, 1999.
9. Dowad, T. PAWS: Personal action wireless sensor. *Personal and Ubiquitous Computing*, 10, 1-3, 173-176. Jan. 2006.
10. Fleck, M., Frid, M., Kindberg, T., O Brein-Strain, E., Rajani, R., and Spasojevic, M. From Informaing to Remembering: ubiquitous systems in interactive museums, Pervasive Computing, 1(2), 13-21, 2002.
11. Gamma, E., Helm, R., Johnson, R.., and Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*.
12. Hallaway, D., Feiner, S., and Höllerer, T. Bridging the gaps: Hybrid tracking for adaptive mobile augmented reality. *Applied Artificial Intelligence*, 25, 477–500, 2004.
13. LaMarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I., Scott, J., Sohn, T., Howard, J., Hughes, J., Potter, F., Tabert, J., Powledge, P., Borriello, G., and Schilit, B. Place Lab: Device positioning using radio beacons in the wild, in *Proc. of PerCom'2005,* 116–133, 2005.
14. Li, Y., Hong, J. I., Landay, and J. A. Topiary: a tool for prototyping location-enhanced applications. In *Proc. of UIST'2004*.
15. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L. and López-Jaquero, V. USIXML: A language supporting multi-path development of user interfaces. In *Proc. of EHCI/DSVIS*, LNCS, 3425, 200–220, 2004.
16. Long, S., Kooper, R., Abowd, G. D., and Atkeson, C. G. Rapid prototyping of mobile context-aware applications: The Cyberguide case study. In *Proc. of MobiCom'96*, 97–107.
17. Okuda K., Yeh, S., Wu, C., Chang, K., and Chu, H. The GETA sandals: A footprint location tracking system. In *Proc. of LoCA'2005*. LNCS, 3479, 120–131, 2005.
18. Orr, R., Orr, R. J., and Abowd, G. D. The smart floor: A mechanism for natural user identification and tracking. In *Proc. of CHI 2000*, ACM Press, 1–6, 2000.
19. Paciga, M., and Lutfiyya, H. Herecast: an open infrastructure for location-based services using wifi. In *Proc. of IEEE Conf. on Wireless And Mobile Computing, Networking And Communications*, 4, 21–28, 2005.
20. Pascoe, J. The stick-e note architecture: Extending the interface beyond the user. In *Proc. of IUI'1997*, 261–264, 1997.
21. Pascoe, J., Ryan, N., and Morse, D. Using while moving: HCI issues in fieldwork environments.*TOCHI* 7(3), 417-437, 2000.
22. Paternò, F. Model-Based Design and Evaluation of Interactive Applications. Applied Computing. Springer-Verlag, 1999.
23. Patil, A., Munson, J., Wood, D., and Cole, A. Bluebot: Asset tracking via robotic location crawling, *Computer Communications*, 31(6), 1067–1077, 2008.
24. Priyantha, N. B., Chakraborty, A., and Balakrishnan, H. The Cricket Location-Support System, in *Proc. of ACM Conf. on Mobile Computing and Networking*. ACM Press, 32-43, 2000.
25. Ran, L., Helal, S., and Moore, S. Drishti: An integrated indoor/outdoor blind navigation system and service, in *Proc. of PerCom'2004*. IEEE Computer Society, 23–32, 2004.
26. Randall, J., Amft, O., Bohn, J., and Burri, M. Luxtrace: indoor positioning using building illumination. *Personal and Ubiquitous Computing*, 11(6), 417–428, 2007.
27. Ravi, N., Shankar, P., Frankel, A., Elgammal, A., and Iftode, L. Indoor localization using camera phones in *Proc. of the 7th Workshop on Mobile Computing Systems and Apps.*, 2006.
28. Salber, D., Dey, A. K., and Abowd, G. D. The context toolkit: Aiding the development of context-enabled applications. In *Proc. of CHI'99*, 434–441, 1999.
29. Sharifi, G., Deters, R., Vassileva, J., Bull, S., and Röbig, H. Location-Aware Adaptive Interfaces for Information Access with Handheld Computers. LNCS 3137, 305-328, 2004.
30. Tesoriero, R., Gallud, J. A., Lozano, M. D., and Penichet, V. M.R. Tracking autonomous entities using RFID technology. *IEEE Trans. on Cons. Elec.* 55, 650–655, May 2009.
31. Trevisan, D. G., Gemo, M., Vanderdonckt, J., and Macq, B. Focus-based design of mixed reality systems. In *Proc. of Tamodia'2004*, pp. 59-66.
32. Vanderdonckt, J., and Bodart, F. Encapsulating knowledge for intelligent automatic interaction objects selection. In *Proc. of CHI'93* ACM Press, 424–429, Apr. 1993.
33. Vanderdonckt, J. A MDA-compliant environment for developing user interfaces of information systems. In *Proc. of* CAiSE'05, Springer-Verlag, 13–17, 2005.
34. Want, R., Schilit, B. N., Adams, N. I., Gold, R., Petersen, K., Goldberg, D., Ellis, J. R., and Weiser, M. An overview of the ParcTab ubiquitous computing experiment. *IEEE Personal Communications*, 28–43, 1995.
35. Ward, A., Jones, A., and Hopper, A. A new location technique for the active office. *IEEE Personal Communications*, 4, 42–47, Nov. 1997.
36. Weiser, M., and Brown, J. S. The coming age of calm technology. *Beyond Calculation: The Next Fifty Years of Computing*. Copernicus, 75–85, 1997.