UsiXML extension for Awareness Support (Draft)

Jose Figueroa-Martinez [1], F. L. Gutiérrez Vela [1], Víctor López-Jaquero [2], Pascual González [2]

[1] University of Granada
C/ Cuesta del Hospicio s/n - 18071
Granada, Spain
[2] University of Castilla-La Mancha
CP 02071
Albacete, Spain
jfigueroa@ugr.es, fgutierr@ugr.es, victor@dsi.uclm.es, Pascual.Gonzalez@uclm.es

Abstract. Awareness support in MDA technologies is virtually nonexistent. Furthermore, until recently there was no conceptual model suitable for representing Awareness support in model based architectures. Here, we introduce an extension of UsiXML user interface description language to support Awareness as an information requirement. UsiXML aims to describe multi-context and multimodal user interfaces. The model-based approach of UsiXML makes it a good candidate for integrating *Awareness Support* from the requirements phase to the final user interfaces. It enables Awareness requirements to be traced from the final user interfaces to the tasks and domain entities that generate them, allowing developers to maintain and validate all the Awareness mechanisms provided by the system. This leads not only to a better quality of system developed, but also an organized and traceable development of Awareness mechanisms, easier maintenance and improved user interaction.

Keywords: Awareness, model-based user interface development, requirements

1 Introduction

The interest in model-based development of user interfaces [1] has been expressed through the creation of a W3C Incubator Group [2], which takes into account the standardization of model-based user interface design. One of the most active User Interface Description Languages (UIDL), based on the available tools and the community that supports it, is UsiXML [3], which is a XML based markup language used to describe multi-context and multi-platform user interfaces at different levels of abstraction.

UsiXML, as well as other UIDL, does not support the specification of awareness requirements nor the awareness itself, even though awareness features should be part of the specification of any collaborative system.

In general, Awareness means "the knowledge of what is going on" [4]. Awareness support as part of modern model-based development tools is still limited [5], mainly because the gap of development-oriented conceptual models.

UsiXML is based on the Cameleon Reference Framework [6], which defines UI development steps for multi-context interactive applications. The development steps are Task & Concepts (T&C), Abstract User Interface (AUI), Concrete User Interface (CUI) and Final User Interface (FUI), which represents the operational UI.

The objective of this work is to extend UsiXML step by step, introducing Awareness as a new requirement that can be defined, linked, propagated and validated at all stages of the methodology.

2 Awareness Support in UsiXML

We have created several models which will provide the scaffolding for awareness support in UsiXML. The standard transformation process must be changed in order to generate the new mappings that make awareness support traceable.

First, we demonstrate how Awareness is represented. Next, we show how Awareness is included in a domain, in order to obtain a usable source of data. We subsequently describe how to use Awareness data and how to include it in the software development process.

Generic Awareness Representation

According to Endsley's description of awareness [4], an Awareness of "something" (the existence of a resource, the location of some entity, the duration of a process, etc.) is defined by a set of information elements. "Elements" signify something that the observer can receive and understand, such as location, height, weight, size, speed, etc. These concrete elements are features of the observed entity.

There may be other elements that are created by the understanding of the observer, which we call composite elements. The *composite element* is defined as a composition of *concrete elements* and/or other *composite elements*, joined by a *compositing function*. Furthermore, it may also be possible to obtain projections of the value of some elements in the near future. We call this Awareness Projections (Fig. 1a).

This Awareness Abstract Representation (AAR) can be useful for sharing and reusing generic models of awareness types that are already known. However, for concrete Awareness support a domain-specific awareness representation is required.

Domain specific Awareness representation

In order to define specific awareness types for the entities of a particular domain, these awareness types must be linked to domain class attributes, in order to create usable awareness data sources. We call this combination an Awareness Concrete Type (ACT) and it represents the way to define the supported awareness in a system (Fig. 1a). We should be able to define many ACTs, some of them from the same AAR. For example, *Group Location Awareness*, *User Location Awareness*, *Pointer Location Awareness*, are all conceptually equal to *Location Awareness*.

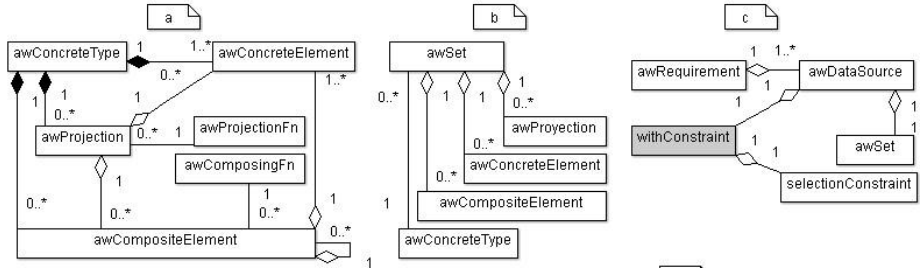

Fig. 1. a) ACT model. b) Awareness Set model. c) Awareness Data Source Model.

In order to define an ACT we must create a structure (similar to the AAR that the type comes from) which is linked to the observed entity and which has the concrete elements (and composite elements) linked to the corresponding attributes (*composeAttribute*) of the observed entity, as shown in Fig. 1a. It is necessary to extend the Domain Model with the entities shown in Fig. 2a in order to define composed attributes.

Awareness Requirement representation.

Once awareness types have been defined for the domain, the next step is to use them. For this, some UsiXML model changes are required to help deal with issues such as privacy and disruption. Privacy means that awareness data is only given to the users that need it. Accordingly, a new entity is added to the Context Model (Fig. 2b) to deal with this Runtime Condition (RC). To reduce disruption problems, the system should provide users with only the awareness data they need (Fig. 1 b & c).

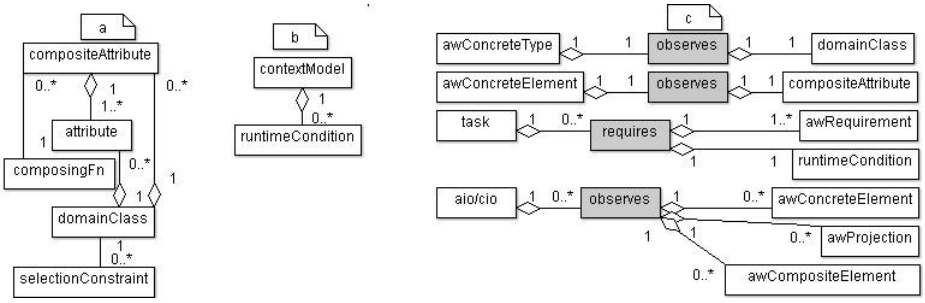

Fig. 2. a) Extension to Domain Model. b) Extension to Context Model. c) Added mappings.

The system should provide all the awareness information required by any user for performing a task. This means that during a task, users must receive the updated data they require, which can be different depending on the runtime context. Our proposal is to add Awareness Requirements (AR) to a task (Fig. 2c) by using mappings [7]. An AR is the representation of an awareness information requirement for a specific case or context (RC) which accesses a selected group of instances from the awareness data source (*awDataSource*, *awSet*). Fig. 1 b & c.

3 Conclusions and Future Work

This work presents a model-based approach for developing awareness supported muti-platform and multi-modal UIs based on the UsiXML methodology. The novelty of this proposal is that Awareness is included as an information requirement and is managed at all stages of the methodology, allowing reusability, traceability, verifiability as well as the ability to be processed by software.

The logical separation of awareness as a requirement and user interfaces improves the quality and maintainability of both, and creates new ways to improve the development process and enable automated testing of awareness mechanisms.

The *awareness support* defined as a requirement model opens the door to other forms of abstractions. The knowledge stored in the models can be used in novel ways and offer more advantages to developers and users. Furthermore, some important problems like privacy and disruption related to the awareness support are tackled by using the "runtime conditions" along with the Awareness Requirements, and by using "selection constraints" that clearly select the awareness data sources.

As future work, we plan to improve the transformation process that manages the awareness models through the different steps of the UsiXML software development process. We also plan to include awareness models in other User Interface Definition Languages and development methodologies.

Acknowledgements

This research is financed by the Ministry of Science and Innovation, Spain, as part of DESACO Project (TIN2008-06596) and the Mexican National Council of Science and Technology CONACYT.

References

1. Puerta, A.: A Model-Based Interface Development. In: IEEE Software, vol. 14, No 4, pp. 40—47. (1997)
2. Cantera, J.M.: Model-Based UI XG Final Report. W3C Incubator Group Report 04 May 2010, http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui/ (2010)
3. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López Jaquero, V.: UsiXML: a Language Supporting Multi-Path Development of User Interfaces. In: Proc. EHCI-DSVIS'2004. LNCS, vol. 3425, pp. 200-220. Springer-Verlag, Berlin, Germany (2005)
4. Endsley, M.R.: Towards a Theory of Situation Awareness in Dynamic Systems. In: Human Factors, vol. 37, issue 1, pp. 32—64, (1995)
5. Figueroa Martinez J., Gutierrez Vela F. L., Collazos C. A.: Awareness Models for the Development of Ubiquitous Systems. In Juan Carlos Augusto, Juan M. Corchado, Paulo Novais, and Cesar Analide (Eds.), ISAmI, (72):237-245, Springer (2010)
6. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. In: Interacting with Computers 15,3 (June 2003), pp. 289—308. Elsevier (2003)
7. Montero, F., López Jaquero, V., Vanderdonckt, J., González, P., Lozano, M.D., Solving the Mapping Problem in User Interface Design by Seamless Integration in IdealXML. DSV-IS'2005), Newcastle upon Tyne, England, July 13-15, 2005. LNCS 3941, Springer-Verlag, Berlin, Germany, ISSN: 0302-9743, (2005)