Notational Support for the Design of Augmented Reality Systems

Emmanuel Dubois[1], Paulo Pinheiro da Silva[2], and Philip Gray[1]

[1] Department of Computer Science, University of Glasgow,
17 Lillybank Gardens, Glasgow, G20 6HW, United Kingdom
(emmanuel, pdg)@dcs.gla.Ac.uk
[2] Department of Computer Science, University of Manchester,
Oxford Road, Manchester M13 9PL, United Kingdom
pinheirp@cs.man.ac.uk

Abstract. There is growing interest in augmented reality (AR) as technologies are developed that enable ever smoother integration of computer capabilities into the physical objects that populate the everyday lives of users. However, despite this growing importance of AR technologies, there is little tool support for the design of AR systems. In this paper, we present two notations, ASUR and UMLi, that can be used to capture design-significant features of AR systems. ASUR is a notation for designing user interactions in AR environments. UMLi is a notation for designing the user interfaces to interactive systems. We use each notation to specify the design of an augmented museum gallery. We then compare the two notations in terms of the types of support they provide and consider how they might be used together.

1 Introduction

The integration of digital (virtual) information and actions with the physical (real) world of users through the use of augmented reality (AR) techniques is becoming a crucial challenge for designers of interactive systems. AR is becoming widely used in a number of domains, including leisure [23], maintenance [8], construction and architecture [24] and surgery [4]. Despite the increasing development of AR systems, neither tools nor methods have been proposed specifically for the design of AR systems. Furthermore, AR systems remains largely ad hoc and exploratory. In [5], we proposed a classification space for "Mixed Systems", interactive systems combining physical and digital entities, that identify two kinds of such systems:

- systems that enhance interaction between a user and his/her physical environment by providing additional computer capabilities or data to the physical objects of the environment: these are Augmented Reality systems, AR;
- systems that make use of physical objects to enhance the user's interaction with a computer: these are Augmented Virtuality systems, AV.

The notion of Mixed Reality, introduced by Milgram and Kishino [13], refers to systems that mix digital and physical entities into a single digital representation; for example, the representation of an interior design by merging pictures of a real chair within a 3D graphic model of the room [25]. However, this combination of digital and physical properties or entities is achieved on a monitor, so the perception of the physical world is not direct. As opposed to Mixed Reality, augmented reality approaches developed in the HCI community focus on the integration of computational capabilities with physical objects involved in the user's interaction. Users benefit from complementary computer capabilities when interacting with their usual physical tools and objects. These HCI approaches are user- and interaction-centred, although they differ in the aspects used to characterise an interaction. Four distinct aspects that may have an influence on the user's interaction with AR systems are identified in the literature: 1) Type of data provided to the user [2,8,14]: it may be textual, 2D or 3D graphics, gesture, sound, speech or haptic data; 2) Potential physical targets of enhancement to combine physical and digital data [11]: users, physical objects and the environment are the three main targets identified; 3) Adequacy of the provided data to the task, as well as the location where they are perceivable [1]; 4) Ability of the system to bridge the gap between physical and digital entities [21]. As this research suggests, developing an AR system is different from developing other sorts of interactive system. It is often neither obvious nor easy to design and implement appropriate combinations of physical and digital entities, especially in settings where (i) the user may be mobile, (ii) other artefacts may be manipulated and (iii) the interaction must be sensitive to complex aspects of the context of use. In this paper we focus on the description of such systems: without a means of specifying the features that make an AR system distinctively AR, we cannot communicate or explore design solutions that benefit from these features. We thus present two complementary notations for capturing design-significant aspects of AR systems:

- ASUR, a graphical notation that can be used to describe, characterise and support the analysis of mixed environments; and
- UMLi, a conservative extension of the Unified Modeling Language (UML) for interactive systems.

Each notation enables a designer to construct a model of an AR system. ASUR models identify the key objects and agents in an AR environment along with their physical and informational relationships. UMLi models describe behavioural and structural aspects of the software systems that make up AR systems. In particular, UMLi models include abstract descriptions of user interface presentations implemented by the software systems. These models, and the notations in which they are expressed, offer assistance in several respects:

- Making salient the AR systems-specific characteristics of a design;
- Providing a medium in which to reason about such designs and to communicate them;
- Bridging gulfs between different elements of an AR environment design.

We present the roles that each notation might play in the design of AR environments and their underlying software systems. Furthermore, we discuss the potential benefits of combining both notations to design complete AR systems. The remainder of this paper is organised as follows. First we describe the Mackintosh Project used as a case study in this paper. The next two sections describe ASUR and UMLi, and apply both notation to the case study. We then compare the modelling capabilities of ASUR and UMLi for AR systems. Finally we conclude the paper with a brief discussion of future developments.

2 The City Project Scenario

As a vehicle for introducing and comparing ASUR++ and UMLi, we use an example taken from the City Project, a project developed within the Equator consortium [7]. Based on the work of Charles Rennie Mackintosh, a Glaswegian architect of the early 1900's, the City Project has been exploring the augmentation of the permanent Charles Rennie Mackintosh Interpretation Centre, a gallery situated in the Lighthouse, an architecture and design centre in Glasgow, containing exhibits related to Mackintosh's life and work. The aim of this part of the project is to study the impact of combining multiple media to support visitors' activities, especially collaborative activities involving users in the real museum interacting with users exploring a digital version of the same museum ("co-visiting"). For the visitor to the real museum, the system being created is aimed at providing visitors with digital information tailored to visitor's current context. This information tailoring mainly relies on tracking visitor's motions in the museum and location of the exhibits. Visitor activities are thus embedded with computational capabilities. To do so, the Lighthouse has been equipped with a radio-frequency localisation system that gives the location of the visitors. There are several services that will be provided by the system in the Lighthouse. In this paper, we consider only the *FollowSelectedPath* service offered to visitors of the Mackintosh Interpretation Centre. This service provides AR support to guide visitors through a pre-defined path of exhibits. A *Path* is composed of an ordered set of Exhibits. Each *Exhibit* is at a *Location* inside the museum. In addition, the service assumes that the *Visitor* following a predefined path is already connected to the system. Under these conditions, the *Visitor* get information related to:

- The *Path* to follow: it consists of textual directions and distances separating the current position of the *Visitor*from the next *Exhibit* of the followed *Path*.
- The *Exhibits*: once the *Visitor* reaches the next *Exhibit* of the path s/he is following, the system provides her/him with information about the *Exhibit* using specific *Media*, e.g., *Image*, *Video*, that may not be perceivable in the museum (e.g. building material, previous exhibition locations, etc.)

Technically, the *Visitor* uses a PDA to perceive both kinds of information and to confirm the visit to an *Exhibit* before getting directions to the next *Exhibit* on the*Path*. Although inspired by the City Project, it is important to note that

this example does not represent the design of an existing system nor is it a history of an actual design development. Rather, we have chosen this scenario because it represents a realistic design problem (viz., the design brief is a real one). However, the goals of our example scenarios don't correspond to the goals of the City Project and the design alternatives presented below are our own and don't correspond to any that have been developed during the City Project. To differentiate our example from the actual City Project, we will hereafter refer to it as the Mackintosh Project.

3 ASUR Description of the Mackintosh Scenario

ASUR (Adaptor, System, User, Real object) is a notation designed to address the need for a lightweight notation for describing AR systems. Apart from ASUR, we are not aware of any language or notation well-suited to describe AR systems. When designing AR systems, the physicality of the setting becomes crucial. The designer must consider:

- where objects are located in the physical world,
- how they might move,
- their intrinsic physical constraints, such as size, weight, position, etc.
- what can be modified or digitally enhanced,
- how users perceive, manipulate and perhaps carry objects, etc.

Existing notations, such as UML or CTT, are designed to capture properties of computational entities; there is no way to express the potential physical properties of such entities. Therefore, these notations are ill-equipped to capture exactly those aspects that make AR systems special. UMLi is an example of one way of dealing with this problem, viz., extending an existing notation. ASUR illustrates another way, starting from scratch and bringing together exactly those characteristics identified in previous interaction-oriented studies of AR systems and needed to capture AR system-related design issues.

3.1 ASUR Concepts

Firstly, an ASUR description models an interactive system as a set of four kinds of entities, called components: computer **S**ystem (**Component S**), **U**ser of the system (**Component U**), **R**eal object involved in the task as tool or constituting the object of the task (**Component** $\mathbf{R}_{tool}$ and $\mathbf{R}_{object}$), and **A**dapter for Input and Adapter for Output (**Component** $\mathbf{A}_{in}$ and $\mathbf{A}_{out}$) that bridge the gap between the computer-provided entities (S) and the physical world entities, composed of the user (U) and of the real objects relevant to the task (R_{object} and R_{tool}). A relation between two ASUR components may describe a physical collocation (represented by a double line) or an exchange of information (represented by an arrow) between two components. Secondly, an interaction facet consists of an ASUR component and an ASUR relation between this component and the user. Arrows connected to the component U in Figure 1 are examples

of such interaction facets. The second aspect of ASUR is the description of the interaction facets with a set of characteristics of the user's interaction. These characteristics may constitute a basis for the evaluation of usability properties. A more detailed description of ASUR is presented in [6].

3.2 Illustration of ASUR using the Mackintosh Scenario

The diagrammatic representation of the ASUR description is presented in Figure 1. In terms of ASUR, the visitor is the component U, an exhibit is a component R_{objet} observed by the visitor (R_{objet}→U) and the database, represented by V-R_{object}, containing the path and the information related to the exhibits, is included in the component S. An adaptor for output (A_{out}) is required so

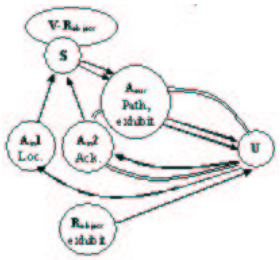

Fig. 1. ASUR Diagrammatic representation of the Mackintosh feature "*Following a path*".

that the visitors perceives the guidance information to follow the chosen path. From this component one relation is connected to the visitor (component U), denoting the transfer of information, related to the path to follow: A_{out}(path→U. Furthermore, an ASUR relation from the component S to the component A_{out} is required because information provided by the A_{out} component is issued by the database (component S): S→A_{out}. Exactly the same reasoning applies to the transfer of information related to the exhibits, leading to the identification of a second adapter for output. However, the scenario stipulates that the PDA has to be used to carry both kinds of information. Consequently, there is only one component A_{out} but two relations from the component S to A_{out} and from A_{out} to the component U, each of them representing respectively the transfer of information related to the path to reproduce and to the exhibits. In addition, an adaptor for input ($A_{in}1$) is required to get the position of the visitor in the museum (U→$A_{in}1$) and transfer it to the computer system ($A_{in}1$→S). Finally, once the visitor has observed an exhibit of the path and potentially read the additional information provided by the system, s/he has to "validate" this step of the path, so that the system can provide direction information to go to the next exhibit of the path. An adaptor for input ($A_{in}2$) is thus required and establishes a bridge between the user's acknowledgment (U→$A_{in}2$) and the state

of the system ($A_{in}2 \rightarrow S$). The fact that the acknowledgment and information visualisation occur on the same PDA is encoded by a double-relation between the components $A_{in}2$ (acknowledgment device) and A_{out} (screen of the PDA): $A_{in}2 = A_{out}$. The same relation exists between the user (component U) and the components $A_{in}2$ and A_{out}, because these two last components are handheld ($U = A_{in}1$, $U = A_{out}$). The second main aspect of ASUR lead us now to characterise the different interaction facets, i.e. the ASUR components and ASUR relations denoting the user's interaction. Four components are involved in this case: the screen of the PDA (A_{out}), its tactile area ($A_{in}2$), the exhibit (R_{object}) and the localiser ($A_{in}1$). The location where the user will perceive and act on the two first components is his/her own hand, since the device is handheld. The visual sense is required to perceive the path and exhibit information provided by the PDA (A_{out}), while physical action, a finger click for example, will be used to acknowledge using the PDA. No information should be shared among users, since different users may have a different path to reproduce. Physical information about an exhibit is perceived on the exhibit itself. This requires a visual sense and this perception must be available for several users at the same time. Finally, the localiser gets information from a tracking area (defined by the technology used for the tracking). When moving in this tracking area, the user will implicitly communicate his/her position to the adaptor. Finally, several users may use this adaptor at the same time. These characteristics are summed up in the Table 1. Concerning the interaction facets, four relations are highlighted in the ASUR diagrammatic description of the situation: perception of the path and exhibit digital information ($A_{out} \rightarrow U$), perception of the physical entity ($R_{object} \rightarrow U$), user's localisation ($U \rightarrow A_{in}1$) and user's acknowledgment ($U \rightarrow A_{in}2$). The first relation carries information expressed in a textual mono-dimensional language, in a frame of reference linked to the visitor so that s/he can read it. The second relation denotes the natural observation of an exhibit: it is based on real 3D language and observed in a user-centred frame of reference. The two last relations correspond to output interaction facets. The user will act with natural 3D actions to either implicitly communicate his/her position to the localiser or click on the PDA to acknowledge. The frame of reference is again user-centred. A summary of the relation characteristics is shown in Table 2.

Table 1. Characteristics of the components that take part in the user's interaction with the system when achieving the Mackintosh project feature "*Following a path*".

Interaction components	**Perceptual/Action location**	**Perceptual/Action sense**	**Information shared**
A_{out} (screen)	User's hand	Visual	Should not
$A_{in}2$ (tactile screen)	User's hand	Physical action	Should not
R_{object} (one exhibit)	Exhibit	Visual	Must
$A_{in}1$ (RF-localiser)	Tracking area	Implicit	May

4 UMLi Description of the Mackintosh Scenario

Model-Based User Interface Development Environments (MB-UIDEs) are a state-of-the-art approach for modelling and implementing running user interfaces from user interface models [18,22]. MB-UIDEs provide models that are effective at capturing user interface functionality [9,16,20], but offer only limited application modelling facilities. Thus an important weakness of MB-UIDEs is in an area of specialism for UML, namely application modelling, while the main strengths of MB-UIDEs align with an area of weakness for UML, namely user interface modelling [19]. Several researchers have investigated the integration of interface modelling techniques with UML. For example, [10] discusses how interface modelling constructs, influenced by those in the TACTICS system, in particular relating to the description of tasks, might be incorporated into UML. A more recent paper [12] assesses several UML models for use in interface modelling, comparing them with a collection of specialist interface modelling notations. In [17] it is suggested how several UML models, in particular class diagrams and use case diagrams, can be used in conjunction with the CTT task model for user interface modelling. In UML for Interactive Systems (UMLi) [19], tasks are modelled using extended activity diagrams rather than through the incorporation of a completely new task modelling notation into UML. Wisdom [15] is probably more mature than the proposals in [12] and [17], in that the relationship between use cases, tasks and views are considered in the paper. UMLi also addresses these relationships but introduces fewer new models into UML and addresses more thoroughly than Wisdom the relationship between tasks and the data on which they act. Overall, the emphasis in Wisdom is probably on earlier parts of the design process than UMLi. Wisdom models tend to be more abstract than those produced using UMLi, but too abstract to generate running user interfaces. Therefore, UMLi is one approach used in this paper as a framework to incorporate AR systems facilities, leading in this way to the development of the first model-based development environment for AR systems. Finally, many aspects of an interactive system can be described by models. Therefore, many models may be combined together when describing an interactive system. However, if the intention is to build UI models that can be used, for instance to generate running user interfaces, two kinds of models should be included: **structural models**, that is mainly *Domain*

Table 2. Characteristics of the relations forming the different facets of the user's interaction with the system when achieving the Mackintosh project feature "*Following a path*".

Interaction facets	Concept	Concept Relevance	Representation Language	Representation Frame of Reference
A_{out}→U	Path and Exhibit	High	1D, textual	Visitor
R_{object}→U	Exhibit	High	3D, real	Visitor
U→$A_{in}1$	User's location	High	3D, real	Visitor
U→$A_{in}2$	Acknowledgment	Medium	3D, real	Visitor

models and *Presentation models* and **behaviour models**. We now present more precisely the domain, presentation and behaviour models of the system and illustrate them within the Mackintosh scenario, in order to explain how UMLi can be used to model the supporting system of an AR environment.

Structural Models The UMLi class diagram represents a schema for the domain of the functional core of the Mackintosh system used to support the computational capabilities required to provide information to visitors. The classes identify the elements of the Mackintosh project previously described in the paper. To address the presentation models, user interface (UI) diagrams are introduced in UMLi to model abstract presentations of user interfaces. As explained in [16], UI diagrams are an alternative notation for class diagrams, providing additional support for interaction classes, which are the classes representing widgets. Namely, the diagram provides visual representation for containment between interaction classes and visual identification for the main role that an interaction class is playing in a particular user interface. Six UMLi constructors for UI diagrams represent different roles of interaction classes:

- **FreeContainers** (dashed cubes) are top-level interaction classes that cannot be contained by any other interaction class (e.g., a top-level window);
- **Containers** (dashed cylinders) provide a grouping mechanism that brings together interaction classes other than FreeContainers (e.g., a frame within a window);
- **Inputters** (downward triangles) receive information from users;
- **Displayers** (upward triangles) send information to users;
- **Editors** (upward rhombi), exchange information in two-ways ;
- **ActionInvokers** (right pointing arrows) receive instructions from users.

In the Mackintosh Project, the UI diagram in Figure 2 represents the *ExecutePathUI FreeContainer*, which is the abstract presentation model of the *FollowSelectPath* service UI. There, *ConfirmExhibit* is the *ActionInvoker* where visitors confirm they have reached the exhibit and *Quit* is the *ActionInvoker* used to finish the service, returning to other functionalities of the system. *Direction* and *Info* are *Displayers* describing the route to the next exhibit of the path and presenting further information about the last reached exhibit. Finally, *GetLocation* is an *Inputter* receiving information about the location of the visitors. As can be observed, *ExecutePathUI* represents relevant design decisions concerning the user interface of the service, avoiding early commitment to concrete properties of the interface. For instance, there is no specification of which kind of widget is going to implement each interaction class. Thus, the *Info Displayer* used to visualise objects of several media (using a PDA during the visit or visiting the digital version of the museum) may be implemented by more than one widget. Regarding AR systems, the *GetLocation Inputter* exemplifies an interesting kind of support that UMLi can provide to this category of interactive systems. Indeed, the *GetLocation Inputter* explicitly represents the localisation system mentioned. Thus, due to its simple mechanisms of abstraction, the UI diagram

provide an appropriate description of how the localisation system, which is a component system of the AR system, interfaces with the rest of the AR system of the Macintosh Project. The UI diagram in Figure 2, along with the domain model, describe the structural properties of the AR system. A behavioural description of the *FollowSelectPath* service is required to complete its specification in UMLi.

Behaviour Models Task models are typically used for modelling interactive system behaviours in MB-UIDEs [16,22]. However, the notion of task, as conceptualised in the MB-UIDE community, is represented by use cases and activities in UMLi [16]. Using use cases and their scenarios, designers and expert users can elicit user interface functionalities required to allow users to achieve their goals. Using activities, designers can identify the possible ways to perform actions that support the functionalities elicited using use cases. Therefore, the mapping of use cases into top-level activities describes a set of interface functionalities similar to that described by task models in other MB-UIDEs. In the Mackintosh Project, the *FollowSelectedPath* service is represented by a use case and visitors are represented by an actor who communicates with the use case. This use case is directly mapped into the *ExecutePath* activity in Figure 3. Furthermore, object flows in activity diagrams, e.g., the vs object of type Visitor and the qt object of type *ActionInvoker*, denote the use of instances of classes to perform actions in action states. For instance, the *pt.startPath()* action state is an invocation to the method *startPath()* of the class Path in the object path. Thus, using object flows, designers can incorporate the notion of state into activity diagrams primarily used for modelling behaviour. In the case of UMLi, the use of instances of interaction objects can also be described by object flows, as in the case of the qt object. However, object flow states, which are rendered as dashed arrows connecting objects to action states, have a specific semantics when used to associate interaction objects to activities and action states. UMLi specifies five categories of object flow states specific for interaction objects described as follows:

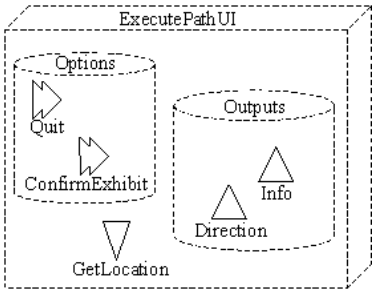

Fig. 2. The UI diagram of the *ExecutePathUI* FreeContainer.

– << *interacts* >> relates primitive interaction objects to action states, which are primitive activities. It indicates that associated action states are responsible for interactions where users are invoking object operations or visualising the result of object operations.
– << *presents* >> relates FreeContainers to activities. It denotes that the associated FreeContainer should be visible while the activity is active.
– << *confirms* >> relates ActionInvokers to selection states. It specifies that the selection state has finished normally.
– << *cancels* >> relates ActionInvokers to composite activity or selection state. It specify that the activity or selection state has not finished normally and that the flow of control should be re-routed to a previous state.
– << *activates* >> relates ActionInvoker to activity, thereby making the associated activity a triggered one, which is effectively started on the occurrence of an event.

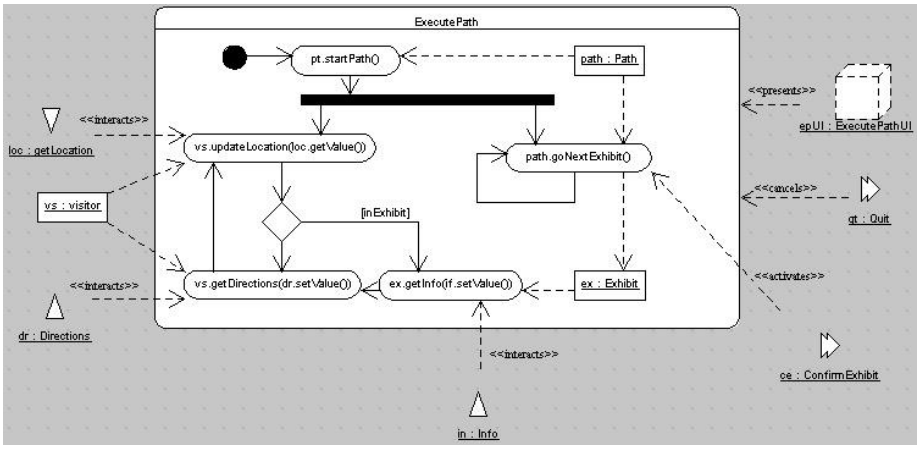

Fig. 3. The ExecutePath activity representing the behaviour of the *FollowSelectPath* service. The *FollowSelectPath* service has been modelled along with other functionalities.

The activity diagram in Figure 3 illustrates the use of most of these interaction object specific object flows. For instance, the *ExecutePathUI* FreeContainer is made visible and the *Quit* ActionInvoker is enabled when the *ExecutePath* activity is active. Then, the *loc* Inputter starts to constantly collect information about the location of visitors producing directions on the *dr* Displayer and exhibit's information on the *in* Displayer. In parallel with this process of producing directions and exhibit information, the service is able to receive a message either from the *ce* ActionInvoker saying that the visitor has reached the next exhibit in the path or from the *qt* ActionInvoker invoking the *ExecutePath* activity. The

service finishes when the system control-flow leaves the *ExecutePath* activity. The following section analyses the differences and similarities of the ASUR and UMLi descriptions of the Mackintosh project and shows how both notations may be combined to support the design process of AR systems.

5 ASUR and UMLi Comparison

5.1 Model-based design of AR systems

UMLi provides on top of UML the ability to express the interface presentation of software systems in an abstract way, identifying the abstract interaction objects required for the user's interaction with the underlying software system. Moreover, the UI diagram allows the composition of abstract interaction objects. Together, this constitutes a step towards an integration of user interface design with underlying system design. In the case of AR systems, however, the user's interaction with the system requires a comprehension of AR environments beyond the specification of windows-based interactions. Indeed, the designs involve the use of physical entities and it has to take into account the behaviour of such entities. ASUR can fill this gap, since it highlights the components required to

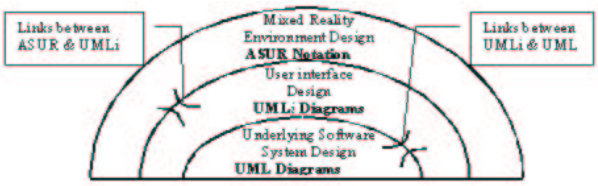

Fig. 4. Layers playing a role in AR systems design and candidate-tools to support these layers.

support the whole human-AR environment interaction, as well as the exchange of information among them, which represents the different facets of the user's interaction with the system. Moreover, ASUR characterises the entities by comparing and taking into account their physical properties. ASUR doesn't support the expression of component refinement and description of software systems of AR environments (i.e. detailed specification of computer-based components); UMLi diagrams, however, provide a solution. Furthermore, ASUR contributes to the specification of UI presentations by specifying those aspects of interaction that relate to the use of physical entities. As a result, we identify 3 layers to consider when designing an AR system (cf. Figure 4):

– **Underlying software system layer**: UML diagrams constitute an approach to design this layer, composed of a set of core components, methods and behaviours.

– **Human-computer interaction layer**: this layer corresponds to the interface presentation specification. UI diagrams proposed by UMLi along with some ASUR concepts help designers to model the presentation of software systems, augmented by other designs tools such as usability studies and guidelines.
– **Human-AR environment interaction**: this is specific to AR systems, in which a part of the interaction relies on physical entities rather than on the software system's interface. ASUR describes this part of the environment and the human-computer interaction in terms of entity-relation models and characteristics. ASUR provides a framework to support the reasoning about different design issues for AR not covered by conventional design solutions for interactive systems.

A model-based AR system design may rely on the specification of properties presented in both ASUR and UMLi notations as illustrated in this paper. The combination of these three layers and bridges among them would link both notations, resulting in a first step towards a Model-Based Design Environment for AR Systems. Thus, between the computer system and the external specification, the bridge is quite straightforward, since UMLi (external specification) is based on the UML diagrams (functional core). Bridging AR-system external specification design and conventional external specification design is required too. We detail these links in the next section.

5.2 Links between ASUR and UMLi

The use of ASUR and UMLi to model the *Following a path* service shows that these notations can be used to describe AR environments. However, the ASUR model in Figure 1 is obviously different from the UMLi models of the same service in the domain models and Figures 3 and 4. In the case of this service, the differences between the models indicate that ASUR and UMLi can support the construction of complementary models of AR environments. In fact, the ASUR model presents an AR-user-centred perspective of the same AR environment that in presented under an AR-system-centred perspective in the UMLi models. A notation for supporting the modelling of AR environments would ideally require a mix of concepts used in both ASUR and UMLi notations. Table 3 is an attempt to identify similarities between constructs used in ASUR and UMLi by comparing their constructs to concepts informally described in the **Concepts** column of the table. Thus, concepts supported in both notations indicate the similarities between ASUR and UMLi constructs. The identification of the similarities of constructs is an indication of how the notations could be used to complement each other to comprehensively support the design of AR environments. ASUR and UMLi also exhibit some mismatches, highlighted in Table 3. Thus, for example, concerning the digital space, a difference between the notations is the absence of refinement of the digital entities in the ASUR notation. Concerning the physical space, another difference resides at the level of the characteristics of the interactions and I/O devices. UMLi does not support the specification of

Table 3. A comparison of ASUR and UMLi implementation of AR-environments concepts.

Context		Concepts	Concepts Representation	
			ASUR Construct	UMLi Construct
A.R. Environment	Physical Space	User	U	Actor
		Input device (ID)	A_{in}	Inputter
		ID interaction charact.	A_{in} charact.	
		Output device (OD)	A_{out}	Displayer
		OD interaction charact.	A_{out} charact.	
		Real tool entity	R_{tool}	
		Real object entity	R_{object}	
		Real entities characterisation	R_{object} charact.	
		Interaction(User-Real tool or object)	R→U, U→R	
		Interaction (Users-ID)	U→A_{in}	<<presents>>
		Interaction (Users-OD)	A_{out}→U	<<presents>>
		Interaction charac.	ASUR relations charac.	
	Digital Space	Interaction (ID-Computer system)	A_{in}→S	Interaction object flow
		Interaction (OD-Computer system)	S→A_{out}	Interaction object flow
		Digital entities	S	Class and Object
		Digital entity properties		Attrib. and Op.
		Relationships between digital entities		Associations + their specialisation
		Goals		Use cases
		Tasks		Use cases + Activities
		User interface presentation		UI diagram

this aspect, while ASUR does, and this aspect constitutes the basis of potential usability analysis, discussed and illustrated in [6].

6 Conclusions and Perspectives

The design of AR systems demands new notational tools to deal with the central role in AR systems, and AR systems design, of the physical properties of the interaction entities and the relationship of physical with informational entities. ASUR is a potentially useful candidate for this role but it doesn't have the capability for describing the user interface(s) that are part of such AR systems. UMLi, on the other hand, is tailored for just this job but, unlike ASUR, it cannot capture the physical properties of components and their relationships with other

entities and with information flows. In our comparison of ASUR and UMLi we identified three levels of design involved in AR systems: human-AR environment interaction, human-computer interaction, functional core. ASUR and other HCI design tools deal with the first two levels. Different design alternatives can be described and subjected to analysis in terms of the physical environment (what moves, what touches), the interaction (perception, action, cognition) and the implementation (sensor deployment, wireless vs. wired). UML and UMLi, on the other hand, offer a method of specifying precisely the behavioural aspects of the interactive system. Of course, ASUR and UMLi offer a way of capturing some of the main features of AR systems, but they will probably not be sufficient on their own. We can expect that additional notations may be needed to capture other aspects of the design (e.g., motion patterns of artefacts and user). We envisage several parallel developments from this point in our work, including **(1)** empirical studies of the use of ASUR and UMLi with realistic AR system design problems (validating this analysis, identifying gaps in our understanding of the requirements for AR notations), **(2)** enhancements to the notations (adding to the expressiveness of each notation, looking at the effects of handling multiple collaborating users and augmented artifacts, dealing with scalability issues), **(3)** exploring further links between notations (generating transformations on descriptions), **(4)** providing tool support for editing but also for linking related aspects of designs, comparing alternative designs, carrying out analyses, and generating descriptions in the other notations (semi)automatically. Also, as we stated at the beginning of this paper, our ultimate goal is to develop a systematic approach to AR system design: a design method. Our exploration of ASUR and UMLi and their links constitutes a starting point towards this aim.

Acknowledgements We would like to thank our colleagues in the Equator Consortium for the inspiration for our scenario. It was the Mackintosh Project that initially stimulated our interest in capturing alternative AR designs. We also thank L. Nigay for her constructive comments on the paper.

References

1. Ahlers, K., Klinker, G., H. Breen, D., Chevalier, P.-Y., Crampton, C., Greer, D., S., Koller, D., Kramer, A., Rose, E., Tuceryan, M., Whitaker, R., *Confluence of Computer Vision and Interactive Graphics for Augmented Reality*, in Presence: Teleoperators and Virtual Environments (Special issue on AR),6, 4: 433-451, 1997.
2. Azuma, R., T., A survey of Augmented Reality, in *Presence: Teleoperators and Virtual Environments*, 6, 4, (1997), p. 355-385.
3. Bernsen, O., Foundations of multimodal representations. A taxonomy of representational modalities, in *Interacting with Computers*, Vol. 6, 4, (1994), p.347-371.
4. Cinquin, P., Bainville, E., Barbe, C., Bittar, E., Bouchard, V., Bricault, I., Champleboux, G., Chenin, M., Chevalier, L., Delnondedieu, Y., Desbat, L., Dessene, V., Hamadeh, A., Henry, D., Laieb, N., Lavallée, S., Lefebvre, J.M., Leitner, F., Menguy, Y., Padieu, F., Péria, O., Poyet, A., Promayon, M., Rouault, S., Sautot, P., Troccaz, J., Vassal, P., Computer Assisted Medical Interventions, in *IEEE Engineering in Medicine and Biology*, 4, (1995), p. 254-263.

5. Dubois, E., Nigay, L., Troccaz, J., Chavanon, O., Carrat, L., Classification Space for Augmented Surgery, an Augmented Reality Case Study, in *Conference Proceedings of Interact'99*, (1999), p. 353-359.
6. Dubois, E., Nigay, L., Troccaz, J., Assessing Continuity and Compatibility in Augmented Reality Systems, to appear in *International Journal on Universal Access in the Information Society, special issue on Continuous Interaction in Future Computing Systems*, (2002).
7. Equator Project Web Site: http://www.equator.ac.uk/
8. Feiner, S., MacIntyre, B., Seligmann, D., Knowledge-Based Augmented Reality, in*Communication of the ACM*, n°7, (1993), p. 53-61.
9. Griffiths, T., Barclay, P. J., Paton, N. W., McKirdy, J., Kennedy, J. B., Gray, P. D., Cooper, R., Goble, C. A., Pinheiro da Silva, P., Teallach: A Model-Based User Interface Development Environment for Object Databases. *Interacting with Computers*, 14(1), p. 31-68, (2001).
10. Kovacevic, S., UML and User Interface Modeling. In *Proceedings of UML'98*, pages 235–244, Mulhouse, France, June 1998. ESSAIM.
11. Mackay, W.E., Fayard, A.-L., Frobert, L., Mdini, L., Reinventing the Familiar: an Augmented Reality Design Space for Air Traffic Control, In*Conf. Proc. of CHI'98*, p. 558-565.
12. Markopoulos, P., Marijnissen, P., UML as a representation for Interaction Designs. In *Proceedings of OZCHI 2000*, pages 240–249, 2000.
13. Milgram, P., Kishino, F., A Taxonomy of Mixed Reality Visual Displays, *Transactions on Information Systems*, E77-D(12): 1321-1329, 1994.
14. Noma, H., Miyasato, T., Kishino, F., *A palmtop display for dextrous manipulation with haptic sensation*, Conf. Proc. of Human factors in computing systems, p. 126-133, 1996.
15. Nunes, N., Cunha, J., Wisdom, A UML Based Architecture for Interactive Systems, In *Proc. 7th Int. Wshp. DSV-IS*, p. 191-205. Springer-Verlag, 2000.
16. Paternò, F., *Model-Based Design and Evaluation of Interactive Applications*. Springer, Berlin, 1999.
17. Paternò, F., Towards a UML for Interactive Systems. In *Proceedings of EHCI2001*, LNCS, pages 7–18, Toronto, Canada, May 2001. Springer.
18. Pinheiro da Silva, P., User Interface Declarative Models and Development Environments. In *Proc. 7th Int. Wshp. DSV-IS*, p. 207-226. Springer-Verlag, 2000.
19. Pinheiro da Silva, P., Paton, N. W., UMLi: The Unified Modeling Language for Interactive Applications, in *Conf. Proc. of UML00*, UK, p.117-132, 2000.
20. Puerta, A. R., A model-based interface development environment. *IEEE Software*,8,p.40-47, 1997.
21. Rekimoto, J., Nagao, K., The World through the Computer: Computer Augmented Interaction with Real World Environments, In Proceedings of UIST'95, 1995.
22. Szekely, P., Retrospective and Challenges for Model-Based Interface Development. In *Conf. Proc. of CADUI96*, pages xxi-xliv, Belgium, 1996. Namur University Press.
23. Szalavari, Z., Eckstein, E., Gervautz, M. Collaborative Gaming in Augmented Reality. In *Symp. Proc. on Virtual Reality Software and Technology*, (1998), p. 195-204.
24. Webster, A., Feiner, S., MacIntyre B., Massie, W. Krueger, T., Augmented Reality in Architectural Construction, Inspection, and Renovation, in *Proceedings of Computing in Civil Engineering*, ASCE, (1996), p. 913-919.
25. Whitaker, R., T., Crampton, C., Breen, D., E., Tuceryan, M., Rose, E., Object Calibration for Augmented Reality, In *Conf. Proc. of Eurographics'95*, Vol. 14, 3: 15-28, 1995.