Domain Specific Methods and Tools for the Design of Advanced Interactive Techniques

Guillaume Gauffre, Emmanuel Dubois, Remi Bastide

IRIT - LIIHS
118, route de Narbonne
31062 Toulouse Cedex 9, France
+33 (0)5 61 55 74 05

{gauffre, emmanuel.dubois, bastide}@irit.fr

ABSTRACT

Novel interactive systems such as Augmented Reality are promising tools considering the possibilities they offer, but no real development methods exist at the moment to help designers in their work. We present in this paper a design method for tightly coupling early interaction design choices and software design solutions. Our work is based on an existing model used for abstract UI design, and introduces a second model dedicated to the software UI specification and the model-based process used to derive one from the other. To achieve this, we present here a framework based on domain specific models and transformations to link them and thus support the development process.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentations**]: User Interfaces - *Theory and Methods.* D.2.2 [**Software Engineering**]: Design Tools and Techniques - *User Interfaces.*

Keywords

Mixed Interactive Systems, Model-Driven Engineering, Domain Specific Languages, Metamodeling, Model Transformations, Design Process.

1. INTRODUCTION

In the past 10 years, a new HCI trend has emerged: traditional "Window, Icon, Menu, Pointing device" interfaces tend to be replaced by new forms of interaction that involve physical artifacts, easily manipulated by users. Augmented Reality systems for example, are interactive systems where the realization of a physical task is enriched by the presence of digital information. Tangible User Interfaces and ubiquitous systems are other forms of interactive systems which merge physical and digital worlds. To refer to these approaches and because they deal with similar concepts and techniques, we regroup them in one term: Mixed Interactive Systems (MIS). Thereafter MIS frameworks have been developed and adopt bottom-up or top-down approaches. Each of them brings consequent advances at different level of abstraction of the design [6] but interlacing them remains difficult to accomplish, thus limiting the coverage of the development process.

As the use of Mixed Interactive Systems increases, elaborating a convenient development process becomes necessary. To cover the different steps of such process, our approach promotes the results gathered in the early design steps and bridges the gap between the abstraction level of these results and the implementation. To do so, we articulate models to progress along the development process and adopt a MDE approach, thus introducing a Domain Specific Language [1] for MIS.

2. MIS ENGINEERING FRAMEWORK

Common processes for HCI development include four steps: requirements gathering, design, implementation and evaluation. Figure 1 presents how our tools cover the first three steps. Task models are one of the major tools to support the requirement step: they are used to describe the sequence of sub-tasks (concerning user's activities, system's activities or interactive activities) in a hierarchical form corresponding to the global system task. The design step can be decomposed into two separate phases: UI design and the underlying software specification. The former step is concerned with user's interaction aspects. It may be linked to requirements gathering by combining users' observation, brainstorming or focus-group to collect user needs, and an interaction model to organize them according to the specificities of MIS [4]: domain objects description, user abilities, physical and digital artifacts, interaction forms. In the latter, design aspects related to the software architecture are considered, using a specific model. The next step is the implementation of the system by using component-based platforms improving flexibility and adaptability.

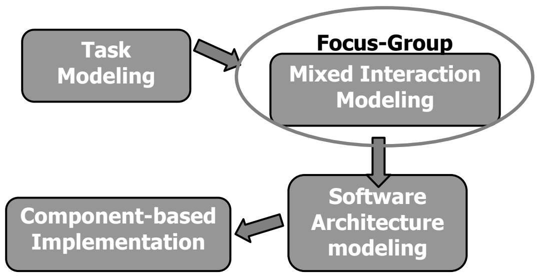
Task Modeling
Focus-Group
Mixed Interaction Modeling
Software Architecture modeling
Component-based Implementation

Figure 1: MIS domain specific process

In this context, rather than modifying the different models we use (task models, interaction models and platform models) to articulate them, we describe a DSL to provide a support to this process:

- Multiple models are required in each step of the development process and one role of MDE is to "promote models to primary artifacts that drive the whole development process" [1]. MDE will facilitate their articulation and permit the elicitation of coherence rules.
- The MIS domain, with regards to their applications in our every day life, produces emergent systems. Elaborating methods to develop them requires to evaluate the adequacy of models and to support their evolution when required. MIS domain is in a phase of empiricism and begins to develop theories; MDE will be a powerful support of this evolution.

3. TWO DOMAIN SPECIFIC MODELS

The DSL we proposed is based on two models:

- ASUR, an existing model which describes the user's interaction with a Mixed Interactive System. It can be used by itself or as mentioned before, in combination with a focus-group.
- ASUR-IL, a complementary model that we introduce to cover the description of the software decomposition and structure. Its aim is to prepare the implementation step by producing a coherent architecture, promoting the interactive forms chosen in a technological perspective.

After an overview of the ASUR metamodel in the next section, we present the ASUR-IL metamodel to enable the collaboration of our two domain specific models.

3.1 ASUR Overview

For a given task, the role of ASUR is to support the description of the physical and digital entities that make up a mixed interactive system and the boundaries among them. ASUR components include adapters (A_{In}, A_{Out}) bridging the gap between both digital and physical worlds, digital tools (S_{tool}) or concepts (S_{Info}, S_{Object}), user (U) and physical artifacts used as tools (R_{Tool}) or object of the task (R_{Object}).

Components can be inter-connected by several kinds of relationships. The major one, *Data Exchange*, is used to describe the kind of data transmitted. In the physical part, they represent the information channels between components, and in the digital part the way the system treats them. The *Representation* link expresses a coupling between a physical component and a digital one in terms of behavior and rendering. Finally *Real associations* express a physical proximity of two physical components and *Triggers* represent an action of one component over another. On the basis of previous works in the domain, design-significant aspects have been identified and added to the model: ASUR characteristics improve the specification of components (*perception/action sense*, *location*, etc.) and relationships (*type of language*, *point of view, dimension, etc.*). By analyzing the characteristics of each element, the model supports the predictive analysis of two properties: continuity and compatibility of interactions.

3.2 ASUR-Implementation Layer: Towards the Implementation Phase

For each ASUR model, i.e. a given mixed interactive task, an ASUR-IL model is associated. The main contribution of this model is to identify the software components and relationships required to implement this specific task. Only the components involved in the interaction part of the system are described. The description of functional parts of the application is out of ASUR-IL scope. This model is also the frontier between Platform Independent Model and Platform Specific Model: it describes the software components involved in the task and their communications, the next step being the transfer to a PSM where each ASUR-IL component will be associated to software component, existing assembly or new ones.

To present this assembly of components, the main concepts of the ASUR-IL metamodel are *Components* and *Data Flows*. A third item *Port,* represents the interfaces between each of them. The correctness of the *data flow* between two *components* is ensured by the value given to the attribute *data type* of each port. There is only one kind of relationships as opposed to components for which the definition follows two principles: correlation with the ASUR components (ASUR *adapters* → ASUR-IL *adapters*, ASUR *System* components → ASUR-IL *Entities*) and roles in the architecture (*Devices*, *APIs*, *Models*, *Controls*, *and Views*).

ASUR-IL *Adapters* in input or output, correspond to the adapters in the ASUR model and group devices and software libraries used to connect physical and digital worlds. *Devices* are used to capture/render data from/to the physical world. They can translate physical phenomenon into digital data. The second part of an *adapter* is an assembly of specific *APIs* which permit to combine several computing facilities to obtain required data, such as ARToolKit, a specific toolkit for Augmented Reality, which, from a captured frame, produces 3D coordinates of the recognized markers.

ASUR-IL *Entities* are the other concepts that make up an ASUR-IL model. They correspond to the digital concepts involved during interaction and identified in ASUR as S_{Tool}, S_{Object} or S_{Info}. They are triplets of three ASUR-IL components called *Models*, *Views* and *Controls*, inspired from the MVC decomposition [7]. *Controls* are in charge of interpreting the physical phenomena translating data from *Adapters* into commands on *Model* parts. *Models* are the entry point to the functional core. They are an abstraction of it, enabling the dialog with the application core. Finally, *Views* are in charge of the computation required to reflect the state of each digital concept on each *Adapter* connected.

3.3 MIS Design Support

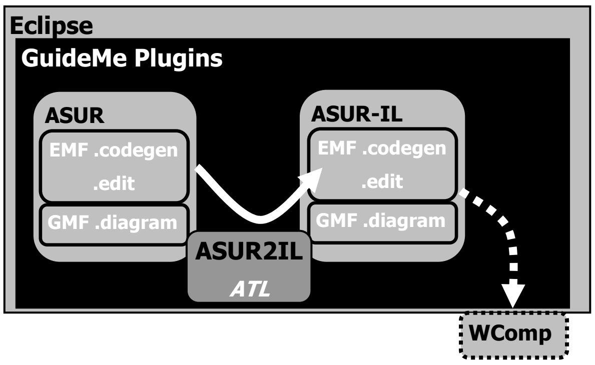

Figure 2 : Tools integration

ASUR has its own editor: GuideMe. It is a graphical editor which can export diagrams as XML files. After its metamodel was defined [3], a second version of the editor has been developed using EMF to separate graphical editing from model manipulation. As mentioned above, ASUR and ASUR-IL are two models required at different steps of a MIS design process. Other models could also be required such as task model for requirements gathering or UML for functional core specification. To support the integration of our two models and further evolution, we adopt an MDE approach and choose to instrument it with tools from the Eclipse Modeling Project (EMP [5]). It enables the creation of dedicated tools for each model with EMF, GMF, and others. Therefore each model can be edited using the corresponding plug-ins in Eclipse (cf. Figure 2).

Thanks to these tools, the designer can manipulate the two models easily. The main challenge is now to couple them by model transformations to rapidly observe the consequences of modifying the description of the interactive situation modeled with ASUR on the software architecture described with ASUR-IL. The next section presents the transformation between ASUR and ASUR-IL and finally introduces the transformation between ASUR-IL and a software component model: WComp [2].

4. DOMAIN TRANSFORMATIONS

In order to implement these transformations, the Atlas Transformation Language (ATL) has been chosen. One of the main reasons is that ATL is now fully integrated in the Eclipse Modeling Project [5] and so ensures us a complete coherence between the different tools. As the targeted platform embeds its metamodel as code and thus using a Model-2-Model engine is actually not possible, we also use a Model-2-Text engine: JET

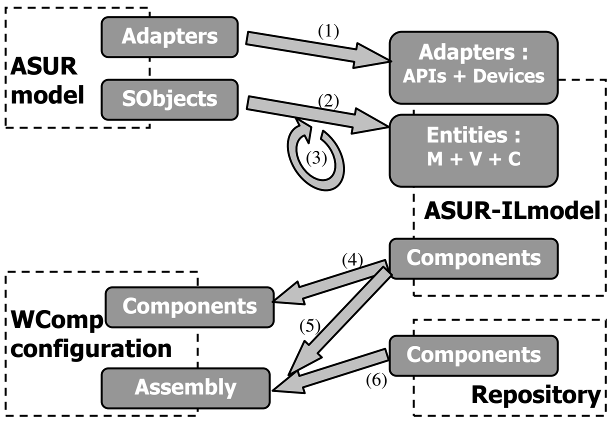

Figure 3 : Specific transformations of MIS process

4.1 ASUR 2 ASUR-IL: Software Modeling Initialization

The goal of this transformation is to prepare the construction of a component-based architecture. ASUR identifies several digital concepts considering their roles in the interaction: this is the left hand side of the transformation. On the right hand side, ASUR-IL is in charge of describing the different kinds of components involved in the interactive part of the system, with adequate ports and data flows between them. Major rules were already defined, but not formalized. The goal was to convert them into ATL rules.

Each ATL rule follows roughly the same behavior. By identifying the type of each ASUR component plus the relationships between them, specific matched rules are involved. It consists, for example, in creating for each ASUR *adapter*, an ASUR-IL *adapter* (Figure 3 - 1) containing one default *device*, and a default *API*. Each rule contains imperative code used to interconnect components (Figure 3 - 3) and to factorize common processes. For example, when ASUR digital components are transposed in ASUR-IL (Figure 3 - 2), they trigger the creation of multiple *Views* and *Controls*, after *Models* have been created.

This transformation is the starting point of the software architecture design. From the characterisation of a mixed interactive situation with ASUR, it produces the base of the software architecture. It offers to rapidly design the structure of a concrete system before starting its implementation. This combination enables now to easily support the designers during the crucial phase linking abstract UI design and software UI specification. Following the transformation, designers can extend the specification by additional design decisions before the next step which is to define a component-based model of the system.

4.2 ASUR-IL 2 WComp: Platform Specific Model Definition

Assuming, that during ASUR-IL edition the designers carefully identified each component of the system, they now must be transposed on the platform model. The currently chosen platform is WComp [2] which is dedicated to rapid prototyping of wearable and ubiquitous interactive systems. Considering these goals, this platform allows the creation of assemblies of components with a small granularity and the runtime adaptation to the platform context (i.e. low battery level, devices disconnected, etc.). Its flexibility and its simplicity are the major points to use it.

The definition of this transformation is an on-going work using ATL and JET. It will make the bridge between our PIM (ASUR-IL) and a PSM (an assembly of WComp components), with two goals:

- to create a component by describing the data manipulated and the interfaces associated (Figure 3 - 4), or to identify a component in a repository (Figure 3 - 6) of already defined components from older projects or standard APIs,
- to manage the assembly of components (Figure 3 - 5) i.e. establishing the connections between each components in accordance with the ASUR-IL model.

Once this transformation is realized, it will be possible to offer designers a range of tools from interaction design to implementation. It will help to rapidly experiment with designed interactive situations from the ASUR results to the WComp assembly of components dedicated to MIS. To illustrate the kind of process it will create, we next describe our tools on a particular case study.

5. TUI FOR MUSEUM EXHIBITIONS

The goal is to design innovative interactive situations in the context of museum exhibitions. Our work is to design solutions promoting knowledge transmission and entertainment in a science museum for particular themes: in this case the species evolution. By using this approach, we can rapidly experiment advanced interaction and adapt them to other themes by reusing components.

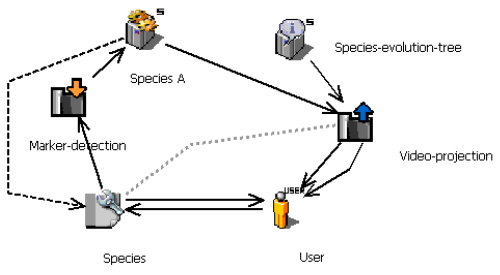

Figure 4 : ASUR model for evolution-tree construction

The current project aims at proposing to visitors to discover species evolution by elaborating an evolution tree based on phylogenetic criteria. Adopting MIS in that context offer the opportunities to manipulate physical objects and to enlarge the experience by digital rendering (video, 3D, sound, etc.). To elaborate the evolution tree, the user manipulates physical representation of species (a frog, a crocodile, etc.) to add them to the tree which is rendered by video on the interactive space with related phylogenetic criteria. The first solution (Figure 4) uses marker-based detection to capture tangible objects (species) and video projection to report the data.

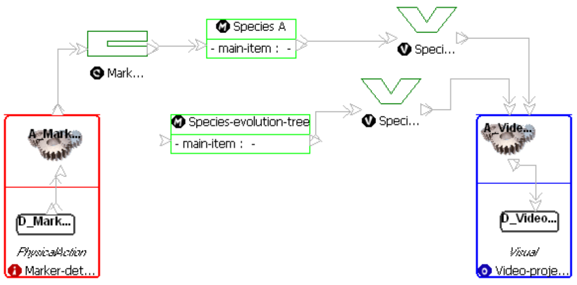

Figure 5: Asur2IL transformation result

Figure 5 shows the ASUR-IL model resulting from the asur2il transformation. The designer can now extend the model: Figure 6 shows an insertion of another camera to modify the marker detection and a specification of the rendering APIs.

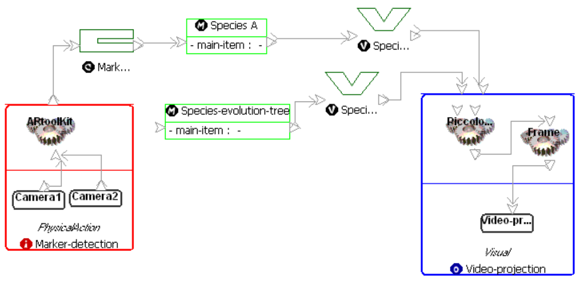

Figure 6: ASUR-IL model extended

To illustrate the dependencies between the two models, we can focus on the case the museum visitors wish to see the evolution tree. It results (Figure 7) by the insertion of an A_{Out} in the ASUR model and in the ASUR-IL model (only one *view* is used because the same interaction modality is used).

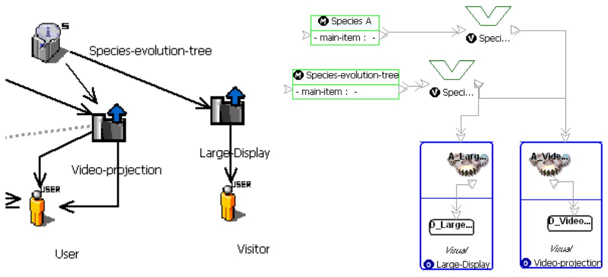

Figure 7: Models evolution

6. CONCLUSION AND FUTURE WORKS

This work is a step toward the definition and instrumentation of a design process for Mixed Interactive Systems. This process will permit to increment on the designed solution until obtaining a convenient degree of usability. The advances presented here, ASUR-IL model and related transformations, offer to rapidly navigate between the abstract design of innovative interaction techniques and their concrete realizations. The Domain Specific Language developed is an efficient tool for promoting the characteristics issued from the user-centered design, into the crucial phase of implementation. As this approach uses models as primary artifacts, thanks to the MDE tools, each level of abstraction defined in the development process embeds properties standing for the usability of the interactive system.

ASUR model defines some properties related to the quality of the interaction between a user and a mixed environment. Our goal is to plainly integrate them among the entire process, to finally evaluate their evolution during each cycle of the process. Further work will aim at identifying additional properties, relevant at the software design level (ASUR-IL) such as computing time or hardware constraints, and structuring their impacts on the remaining design steps of our process. It will increase the ability to evaluate the quality of each interactive situation.

Another perspective is to study the feasibility of reverse transformations between each step and their impact on the higher levels of abstraction. In Figure 7, what would be the impacts of applying a reverse transformation from the modified ASUR-IL model to the ASUR model?

Finally, we focus here on specific models for MIS. To make possible the development of concrete systems, others aspects must be included: collaboration with business models for the connection with the functional core, interactive modalities ontology to support the choice of specific devices and APIs. In this way, we planned to describe, in ASUR-IL, the behaviour of the components using dialog models (State charts, Petri nets, etc.).

As already mentioned, the MDE approach is very helpful to articulate and transform models. However, it appears that designing MIS may rely on a lot of models and maintaining the coherence among all of them may be difficult. The management of this combination of models and transformations need to be investigated to better assess the usability of the MDE approach for a MIS development process.

7. REFERENCES

[1] Bézivin, J., Jouault, F., Kurtev, I., Valduriez, P.: *Model-based DSL frameworks*. 21st ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, Portland - USA (2006)

[2] Cheung, D.F.W., Tigli, J.Y., Lavirotte, S., Riveill, M.: *WComp: a Multi-Design Approach for Prototyping Applications using Heterogeneous Resources*. In Proceedings of the 17th IEEE International Workshop on Rapid System Prototyping, Chania - Crete (2006)

[3] Dupuy-Chessa, S., Dubois, E.: *Requirements and Impacts of Model Driven Engineering on Mixed Systems Design.* In Proceedings of the conference IDM'05, Sébastien Gérard, Jean-Marie Favre, Pierre-Alain Muller et Xavier Blanc (eds.), Paris - France (2005) 43-54

[4] Dubois, E., Gauffre, G., Bach, C., Salembier, P.: *Participatory Design Meets Mixed Reality Design Models.* In conference Proceedings of Computer Assisted Design of User Interface (CADUI'06), Springer-Verlag, Information Systems Series, Bucarest - Romania (2006) 71-84

[5] Eclipse modeling Project - http://www.eclipse.org/modeling/

[6] Hampshire, A., Seichter, H., Grasset, R., Bilinghurst, M.: *Augmented Reality Authoring: Generic Context from Programmer to Designer.* In proceedings of the 20th conference CHISIG of Australia, OZCHI’06, ACM Press, Sydney – Australia (2006) 409-412

[7] Krasner, G.E., Pope, T.: A cookbook for using the Model-View-Controller User Interface Paradigm in Smalltalk-80. *In the Journal of Object Oriented Programming*, (1988) 26-49