Metamodels Infrastructure and Heuristics for Metamodel-Driven Multi-Touch Interaction

Víctor López-Jaquero, Elena Navarro, Francisco Montero and Pascual González

Laboratory of User Interaction and Software Engineering
Computing Systems Department
University of Castilla-La Mancha
Albacete 02071 - Spain
{victor, enavarro, fmontero, pgonzalez}@dsi.uclm.es

Abstract. Novice users usually find it hard to manipulate models by using traditional Model-Driven Development techniques, because of the gap between the modeling tools and these users' mental models. In this context, multi-touch interfaces emerge as an alternative to make it easier for novice users to interact with the models by using natural gestures and taking advantage from the popularity that touch-based devices have achieved. In this paper, a metamodel infrastructure and a set of heuristics are presented to automatically generate multi-touch visual editors for manipulating models. The editor generated is driven by a metamodel that also prevents the user from creating not valid models. These heuristics have been validated while developing an environment for novice users, such as psychologists or physiotherapists, for the treatment of people with Acquired Brain Injury.

Keywords: model-driven development, multi-touch interaction, heuristics, Acquired Brain Injury (ABI)

1 Introduction

The exploitation of models during the software development process is a valuable tool for stakeholders to convey their ideas about design, needs or requirements of the system-to-be. Developers, designers or software architects are used to employ graphical node-link notations for the manipulation of models. UML is a clear example of this approach which is widely used, for instance, to specify class diagrams used at different stages of the development process.

However, it is frequently the case that users and/or clients of the system-to-be are required to manipulate models that are used during the development of the system to convey their expertise in the problem domain. This scenario can arise easily in the user-centered design, where the final users of the application are involved in the development. This can be a challenging, or even overwhelming, task for them as they usually do not have the required abilities to tackle modeling activities. This has been the problem we had to face during the development of HABITAT [24], a system to support the relearning process of people with Acquired Brain Injury (ABI). One of the main cornerstones of this system is its functionality to design new types of relearning

activities so that the relearning process can be fully customized according to the specific needs of the people with ABI. During the development of this functionality, the exploitation of models emerged as a suitable solution. However, our users (psychologists, speech therapists, physiotherapists, etc) did not have the necessary abilities for manipulating models, although they did have the knowledge we needed about the problem domain. This was the challenge we had to face with the ideas presented in this work: can we provide users, who are novice users, with a tool for manipulating models that hides the complexity behind this task? The solution to overcome this complexity has been to exploit the benefits provided by the integration of two well-known approaches: Multi-touch interfaces [23] and Model-Driven Development (MDD, [13, 31]).

Multi-touch interfaces provide users with attractive and innovative facilities for the manipulation of applications by means of touch gestures. Thus, the interaction can be performed in a natural and intuitive way. Several works, such as [9][16], have shown in the experiments performed that this approach is actually suitable for novice users, as they perceived the interaction with multi-touch applications as more attractive and interesting than traditional applications. Therefore, its exploitation in this work emerged in a natural way, since cooperation with novice users is our main goal.

MDD is not only becoming increasingly popular among researchers, but also among practitioners. It has proved to have a positive influence on the reliability and productivity of the software development process due to several reasons, such as, the exploitation of techniques for the automatic generation of code or the use of models as drivers of the development process. Both reasons led us to its consideration in this work; first to introduce the necessary facilities for the manipulation of models and second to generate model-manipulation multi-touch interfaces in an automatic way. The generation of multi-touch interfaces is performed by automating a set of heuristics, which are presented in this work, that exploit the structure and semantics of the primitives used for domain modeling by using *Ecore* metamodels [11].

The paper is organized as follows. Section 2 provides an overview of ABI, by describing the target population and how this work was conducted. Section 3 presents and analyzes the related previous work. Section 4 describes the metamodels infrastructure to support our proposal. Section 5 describes the heuristics developed. Section 6 presents the initial results of the conducted exploratory evaluation. Finally, Section 0 rounds off the paper by presenting the conclusions drawn and some future work.

2 Case study: treatment of Acquired Brain Injury

People with Acquired-Brain Injury (ABI) have suffered “damage to the brain that occurs after birth and which is not related to congenital disorders, developmental disabilities, or processes that progressively damage the brain” [32]. There are several causes of ABI, such as cerebral vascular pathology, skull-brain trauma due to accidents, meningitis, brain tumours, etc. Therefore, it can be stated that just about everybody is exposed to this risk in its daily life. Cases affected by this disability are becoming increasingly common. According to the JCCM Health Council [10] 4 out of every 1000 persons suffer some kind of ABI at some time in their lives. Although

people of all ages can be affected, it is more frequent among the younger and older members of the population, as they are more prone to accidents.

According to the experts, the process of integral ABI relearning must include cognitive treatment, in addition to physical and occupational therapies. It should also be emphasized that ABI associations, such as ADACE, which we have collaborated with in different projects, highlight that ABI victims should be provided with a proper treatment as soon as possible, since there is increasing evidence of its effectiveness during the first stages after injury [8]. However, identifying the proper treatment for each person is a difficult and time-consuming task, since brain injury has dramatically varied effects and no two people can expect the same resulting difficulties. This means that an individualized relearning process must be identified for each person. In this context, providing specialists with tools to create and customize the activities and tasks that they use in the processes of recovery [22] is a must. This is the aim of our system: HABITAT [24].

HABITAT enables ADACE specialists to create relearning activities by instantiating the implemented relearning patterns [22]. These relearning patterns were validated by the specialists and were put into practice thanks to the implementation we made in HABITAT. However, the specialists highlighted the need to customize these patterns and this led us to define the relearning pattern metamodel described in Section 4.1 and its implementation in HABITAT. This metamodel had to be instantiated and used by the specialists to define and modify the special relearning patterns they needed. Nevertheless, as these people were not used to manipulating models by using node-link representations, the alternative was to provide them with multi-touch User Interfaces (UI), which are automatically generated by means of the heuristics presented in Section 5 and using the metamodels infrastructure offered in Section 4.

3 Related work

The generation of UIs or visual metaphors to edit models is not a new trend. All case-tools offer a means of manipulating models by providing a visual notation. Nevertheless, generating a UI to manipulate a domain model is not commonly available. One example of generation of UIs out of a domain model is the Graphical Modeling Framework (GMF, [12]) and Eclipse Modeling Framework (EMF, [11]). By using EMF a developer can create a domain metamodel by means of *Ecore* and then generate a graphical editor by using the GMF framework. The features of this editor are specified by means of a set of models. Another example is Executable UML [19] which aims at generating UIs out of a UML specification. The UIs generated are for standard desktop application, and the interaction is mostly based on *drag&drop* interaction in a tree. The author provides some guidelines regarding how UI is generated, the so called *interactive manifestations*. This approach is not intended for metamodel manipulation, but introduces interesting ideas regarding the generation of UI out of UML models. Nevertheless, the generated UI is not appropriate for novice users and the user has no feedback regarding multiplicities in the specification of the cardinalities of the relationships. Another approach pursuing the generation of UI out of object-oriented specifications is *Naked Objects* [26]. In Naked Objects, the

applications are specified solely by using domain entity objects. A direct matching between domain objects and presentation is proposed. Unfortunately, the heuristics to generate the presentation are not described. As for Executable UML, this approach is not aimed at manipulating metamodels, but at supporting whole application user interface generation. In Naked Objects, the presentation cannot be customized; therefore it cannot be adapted to different user skills or preferences. There is no guidance regarding the order the tasks should be carried out, the target platform is desktop applications and it is not designed for novice users.

There are also other similar approaches from the human-computer interaction community. Model-based User Interface Development Environments (Mb-UIDE) [27] provide a mechanism to design the UI by means of a number of declarative models, which are latter translated into code directly executable on a specific platform or into an intermediate language (usually XML-based). Mb-UIDE has been in use since the beginning of the 90s and it is becoming increasingly integrated into the MDD approach [34].

In Mb-UIDE, the domain model represents the information required by the user to carry out the tasks through the UI. To express these models, different notations have been used, but undoubtedly, the most commonly used are entity-relationship notation and UML class diagrams. Some of the MB-UIDEs using class diagrams are OVID [29], Janus [2], AME [17], Teallach [4], OO-H [7] and IdealXML [21]. Two of the Mb-UIDEs that use entity-relationship notation for domain modeling are Trident [5] and Genius [14]. Just-UI [20] is a MB-UIDE that provides a set of patterns to generate a UI out of a domain model for standard desktop applications. Although these approaches are not aimed at generating UI for the manipulation/edition of a domain model, they provide interesting insights into what should be modeled to automatically generate a UI out of models.

We took inspiration from these approaches to identify the key features that should be modeled to generate a UI, including the requirements to have some extra models apart from the domain model to generate usable UIs to manipulate the models. These extra models enable the modeling of, for instance, the aesthetics of the user interface to be generated. These extra models are described in depth in section 4.

Since our goal is to be able to generate a UI for the manipulation of domain models by novice users, with no experience in using software design tools, the proposal used should not be the one used in most CASE tools. Therefore, some research was carried out on the different interaction techniques available. This study showed that multi-touch user interfaces were the most intuitive ones for novice users, such as ABI specialists [22]. The most important reason why ABI specialists chose multi-touch interaction was because they felt it was natural, and also because they were used to the techniques involved in multi-touch smart phones. Nowadays, multi-touch interaction and gestures are being widely used in many kinds of portable and fixed devices [23].

The concept of interaction style refers to the different ways the user can communicate or otherwise interact with an artifact. There are different interaction styles, for instance: command language, form filling, menu selection or direct manipulation. In our context, direct manipulation was chosen because it offers several advantages such as [30]: visually presenting task concepts, it is easy to learn, errors can be avoided more easily and recognition memory, as opposed to cued or free recall

memory, can be emphasized. There are currently many electronic devices in which touch and motion-based gestures are supported and there are de facto standards related to this type of interaction, such as the Apple Human Interface guidelines [1]. These guidelines have been used for the definition of the heuristics presented in Section 5.

4 Metamodels infrastructure

Several approaches have emerged to date describing the guidelines to execute the MDD paradigm. Perhaps, the most commonly known is the Model-Driven Architecture (MDA, [18]), an initiative of the Object Management Group (OMG). MDA promotes the separation of the domain model from the underlying technology to facilitate higher flexibility while designing and evolving software systems. One of the key elements of the MDA initiative is the Meta-Object Facility (MOF, [25]), a four-layer architecture used for the definition of the metamodels and models involved in the software development process, as shown on the right part of Fig. 1.

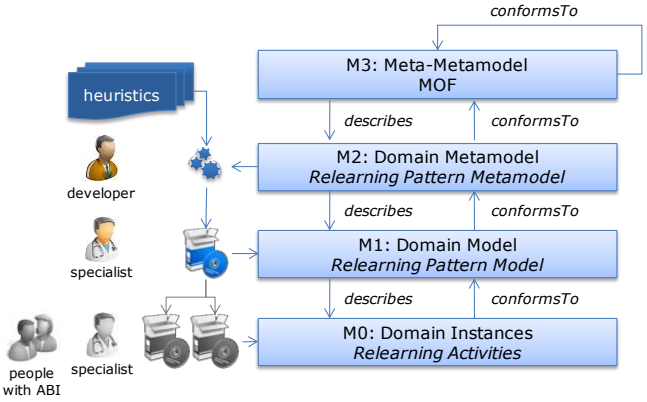

Fig. 1. Stakeholders, software products and MOF architecture in our approach

Fig. 1 shows the workflow of our proposal. The key element is the *instantiation process*: metamodels (M2) are instances of Ecore (M3), models (M1) are instances of metamodels, and domain instances (M0) are instances of models (this level is shown in gray to indicate it is out of the scope of this paper due to space limitations). In this instantiation process each stakeholder, specifically developers and ABI specialists, are focused on their interests and abilities. Developers are responsible for the specification of metamodels and the development of a flexible tool for relearning activity patterns. However, ABI specialists are focused on the definition of relearning activity patterns by using the tool with a high level of usability created by the developers. The UI of this tool was generated automatically by the developers, using the set of heuristics that are presented in Section 5. These heuristics were automated by using XPAND [33], a specialized language for code generation out of Ecore models. Moreover, the editor model was automatically created thanks to the capabilities offered by the Eclipse Modeling Framework (EMF, [11]).

As Fig. 1 shows, the MOF architecture has been used to describe both the domain metamodels (M2) and the models for the systems under development (M1). The domain metamodels are used to describe the core processes and domain concepts that are to be used by developers to convert design into code. According to our case study, the domain metamodel is the relearning pattern metamodel. This architecture has been used as follows:

- *Meta-metamodel (M3) level*. It offers a collection of primitives to define metamodels at level M2, that is, it is a meta-metamodel to describe metamodels. In this proposal, Ecore from EMF was used as meta-metamodel.
- *Metamodel (M2) level*. The elements in this metamodel are used to describe the elements of the model at level M1. ABI specialists demand software for the creation of relearning activities for people with ABI that is customizable enough. Although, these activities are known by specialists, they unfortunately cannot implement them. HABITAT provides a computer-based tool for relearning activity patterns specification for ABI specialists. With this aim, in this M2 level, developers created the domain metamodels that were used jointly with the heuristics presented in Section 5 to create a multi-touch tool in an automatic way. This tool is used by ABI specialists to manipulate the domain model, that is, the model for specifying relearning activity patterns.
- *Model (M1) level* is defined by instantiating the M2 metamodel, which is used to define the relearning patterns that enable the specialists to create relearning activities. ABI specialists work, at this level, with the computer–based tool developed by the developers. ABI specialists create the relearning activity patterns to provide tools for people with ABI. At this level, concrete interaction styles, deficits and resources are specified.
- *Instances (M0) level*. At this level, the instances of the domain model are created. In our case, the relearning activities are defined by the specialists and used by the people with ABI, as instances of the M1 model. ABI specialist instantiate and create different tools for people with ABI where different deficits and interaction possibilities are considered. People with ABI use specific software designed by ABI specialists for their treatment.

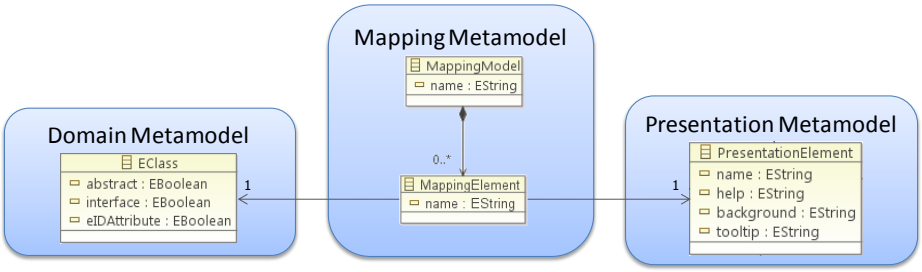

Fig. 2. Metamodels Infrastructure

Although it is not shown in Fig. 1, at M2 level, developers identify and specify three metamodels: domain, mapping and presentation. As it can be observed in Fig. 2,

these metamodels are generic, so that they can be used for any general purpose domain. Next, all these three metamodels are described:

- *Domain metamodel*. This metamodel supports the specification of the facilities and services required for the target domain. As it can be observed, the main element of *Ecore*, *EClass*, has been used to facilitate that any domain metamodel defined as instance of *Ecore* can be used in this proposal. In our case study, developers are aimed at providing an environment for ABI specialists supporting the specification of any number of relearning activities patterns. Later, in Section 4.1 we present the HABITAT relearning activity pattern metamodel as an example of a concrete domain metamodel that can be used.
- *Presentation metamodel*. This metamodel is used to support the specification of the presentation details for each domain metamodel element. Section 4.2 describes the presentation metamodel developed in this proposal.
- *Mapping metamodel*. This metamodel is a mediator entity between domain and presentation metamodels. It relates both metamodels supporting loose coupling between the domain metamodel and the presentation metamodel.

4.1 Domain Metamodel: HABITAT relearning pattern metamodel

At the metamodel level (M2), developers defined a specification useful for relearning activity patterns specification. These patterns constitute the domain to be defined by our novice users, ABI specialists. At the beginning, many ABI specialists, for instance in the ADACE association [22], documented the relearning activities using cards. These relearning activity descriptions had different elements or sections that were analyzed and abstracted away to create the metamodel shown in Fig. 3 (for the sake of clarity, class attributes are not shown in the figure). Purposes and descriptions of the elements of relearning activity patterns are the following:

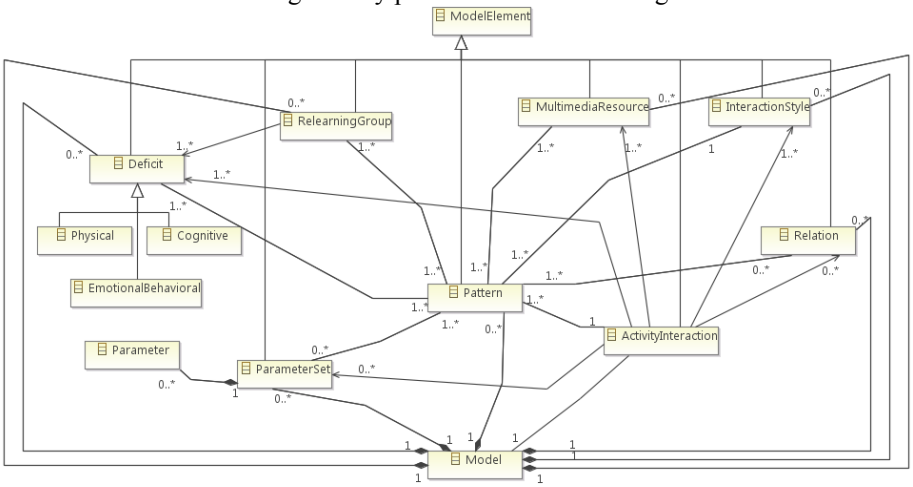

Fig. 3. Relearning pattern metamodel

- *Model*. This is a general-purpose element for organizing the domain metamodel. The definition of this kind of element is a constraint imposed by *Ecore*.
- *ModelElement.* This element was defined to abstract away some attributes that are shared by several metamodel elements.
- *Pattern*. This element represents a relearning activity pattern at a conceptual level, that is, concrete relearning activity patterns will be instances of this element. It represents an abstraction of a type of treatment or activity for people with ABI. It will be instantiated by ABI specialists.
- *Relation*. This element represents the relationships between relearning activity patterns. Few patterns live in isolation. Typically, they introduce new, hopefully smaller and more tractable activities which will lead you to other relearning activities. Or, there may be other patterns that treat the same set of deficits.
- *Parameter*. This element is used to represent the information of each pattern. Instances of this element that ABI specialists can specify are related to evaluation criteria, timing, etc.
- *ParameterSet*. This element represents sets of parameters with a common purpose.
- *Deficit*. Each relearning activity pattern is addressed to treat a specific set of deficits (e.g.: physical, cognitive and/or emotional/behavioral).
- *RelearningGroup*. In order to facilitate the treatment process, different groups were defined by ADACE, each one being characterized by a set of specific deficits.
- *MultimediaResource*. This element is used to associate multimedia resources to relearning activity patterns.
- *InteractionStyle*. This element represents the interaction style that the ABI specialists wants each person with ABI to use when he/she is doing a relearning activity.
- *ActivityInteraction*. This element represents the kind or type of activity that ABI specialist wants to associate with a relearning activity pattern. For instance, association or puzzle activities are instances of this element.

By using the previous elements developers provide ABI specialists with a software tool. Then specialists can document relearning activity patterns. Customization and personalization of this software tool can be achieved by the presentation metamodel that will be described in the next section.

4.2 Presentation metamodel

This metamodel was defined by developers to achieve flexibility in the look and feel of the computer-based tool provided to the ABI specialists. Each domain model will be related to at least one presentation model, that is, an instance of the presentation metamodel illustrated in Fig. 4.

The presentation metamodel elements are described as follows:

- *PresentationModel*: This is a general-purpose element for organizing the presentation metamodel. It has a similar purpose to Model in the domain metamodel (see Fig. 3).
- *PresentationElement*. Each time a domain element is instantiated, a *PresentationElement* will be created to allow the customization of its presentation.

As can be observed in Fig. 4, it helps in describing different parameters, such as background color, tooltip, or label.

- *Font*. This element is used to allow the font customization of domain elements. It is related to alignment, size, etc.
- *Icon*. This element was included to associate each domain element with an icon.
- *Border*. This element is used to support border customization in domain elements.

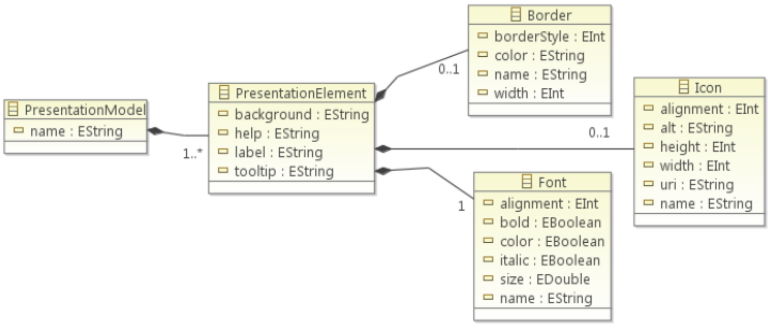

Fig. 4. Presentation Metamodel

Once domain and presentation metamodels were introduced we will describe in the following section the heuristics identified for generating multi-touch UIs that allow novice users to manipulate models.

5 Heuristics for model-driven multi-touch interaction

Heuristics are experience-based techniques for problem solving, learning, and discovery. These methods are used to speed up the process of finding a satisfactory solution, where an exhaustive search is impractical or conditioned in a certain way. Examples of this method include using a "rule of thumb" or common sense. When these heuristics are repeatedly used, and thoroughly tested, they can become patterns. We were able to identify heuristics by applying our previous experience in similar multi-touch interaction UIs developed in collaboration with ADACE. These UIs pursued hiding the complexity usually found when novice users have to manipulate metamodels in a MDD environment.

As shown in Fig. 3, after the developer has specified the domain metamodel, the tool that will be used by the novice users is generated. This tool is generated automatically by means of EMF and XPAND frameworks, and the heuristics proposed in this paper.

The heuristics gathered in this section proved useful for UI development out of a set of models. These UIs were suggested and evaluated by ABI specialists, who were our end-users. Multi-touch and direct manipulation were the interaction styles chosen to support the application. The inputs for the definition of these heuristics were the specialist's requirements. Our activities were driven by the metamodeling principles underlying *Ecore* metamodel.

One of the main elements considered by the heuristics presented in this work is the semantics of the relationships that can established in Ecore. To make easier understanding the heuristics proposed, the semantics of UML relationships [6], and how they can be expressed in Ecore, is described:

- *Association*: An association relationship is a structural relationship between two model elements that shows that objects of one classifier connect and can navigate to objects of another classifier. Even in bidirectional relationships, an association connects two classifiers, the primary (supplier) and secondary (client). This relationship is specified in Ecore by establishing an *EReference* between two *EClasses*. If a bidirectional relationship is needed, then two *EReferences* have to be created, one for each direction, and their *EOpposite* attribute must be set to specify that one is opposite of the other one.
- *Composition*: a composition relationship represents a whole–part relationship, and it is a specific type of aggregation, which is another type of UML relationship. An aggregation relationship depicts a classifier as a part of, or subordinate to, another classifier. A composition relationship specifies that the lifetime of the part classifier is dependent on the lifetime of the whole classifier. To specify this type of relationship in Ecore, an *EReference* between two *EClasses* is created. Besides, its attribute *containment* must be set to true.
- *Generalization*: A generalization relationship denotes that a specialized (child) model element is based on a general (parent) model element. Although the parent model element can have one or more children, and any child model element can have one or more parents, typically is the case that a single parent has multiple children. This relationship type is described in Ecore by setting the attribute *ESuperType* of the child *EClass* to its parent *EClass*.

Other important attributes used while describing Association and Compos*ition* relationships in Ecore are *LowerBound* and *UpperBound*. They are used to describe the cardinality, that is, how many instances from one of the entities is related to each instance in the other entity involved in the relationship.

While the ABI domain metamodel was being developed, and UIs were being discussed with ABI specialists, we identified empirically a set of heuristics correlating the semantics and structure found in the metamodel and the user interaction to manipulate it. These heuristics were gathered in specific scenarios, but we found that they can be reused again and again in other scenarios. In our proposal, multi-touch and motion-based gestures are used to design the following heuristics:

- *Heuristic #1* – [Root]: all the derivation of the user interface starts from the root of the model as this is a constraint imposed by *Ecore*. For instance, *Model* was the selected root for the Relearning pattern metamodel shown in Fig. 2.
- *Heuristic #2* – [Recursion]*:* as the user browses the elements of the domain model to create and manipulate an instance, all the heuristics are applied recursively. That is, the root where all the heuristics are being applied is the current element the user is browsing.
- *Heuristic #3* – [Association (2-directional) - Drag&Drop]. A bidirectional association relationship is used, for example, to specify that a *pattern* can be

related to several *relearning groups*, and that a *relearning group* can be related to several *patterns* (see Fig. 3).

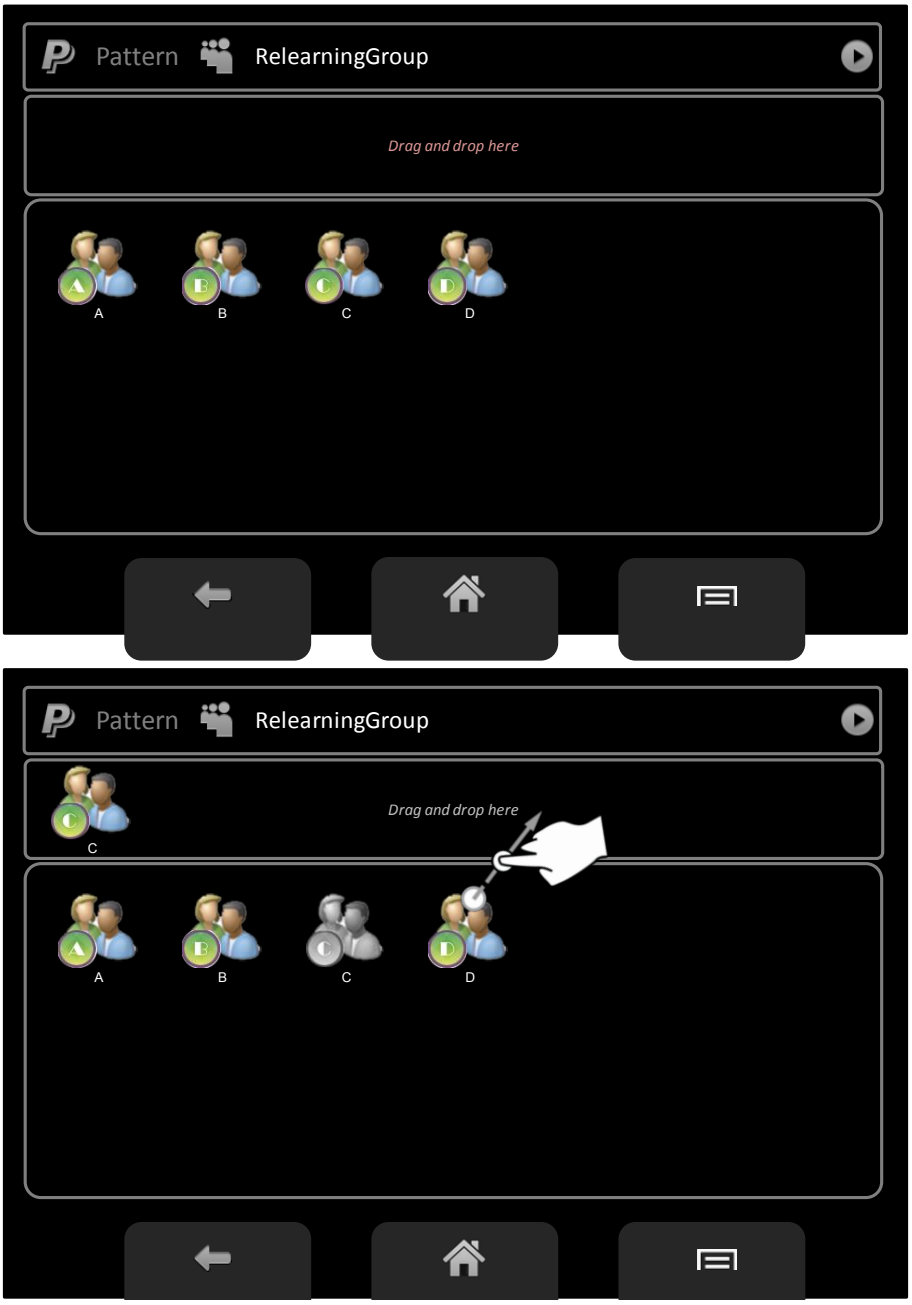

Fig. 5. Generated UIs and gesture according to Heuristic #3

Drag&Drop is the interaction style to be used whenever an association of the domain model has to be manipulated. As shown in Fig. 5, by dragging one instance from the lower part, and dropping it into the upper part, the user can easily create the relationship between a *pattern* and a *relearning group*. To provide more guidance to the user, and prevent errors, feedback related to the number of elements that the user can or must drag is provided. If the user can relate or not the elements (optional relationship), then a suggestion written in grey text is shown in the upper part. On the other hand, if it is mandatory for the user to create the relationship (mandatory relationship) then this text is written in red. Note that depending on the path followed to reach an element in the metamodel, this heuristic represents one direction or the other of the bidirectional relationship. Always the direction represented in the UI is the one whose origin is the current element the user is manipulating.

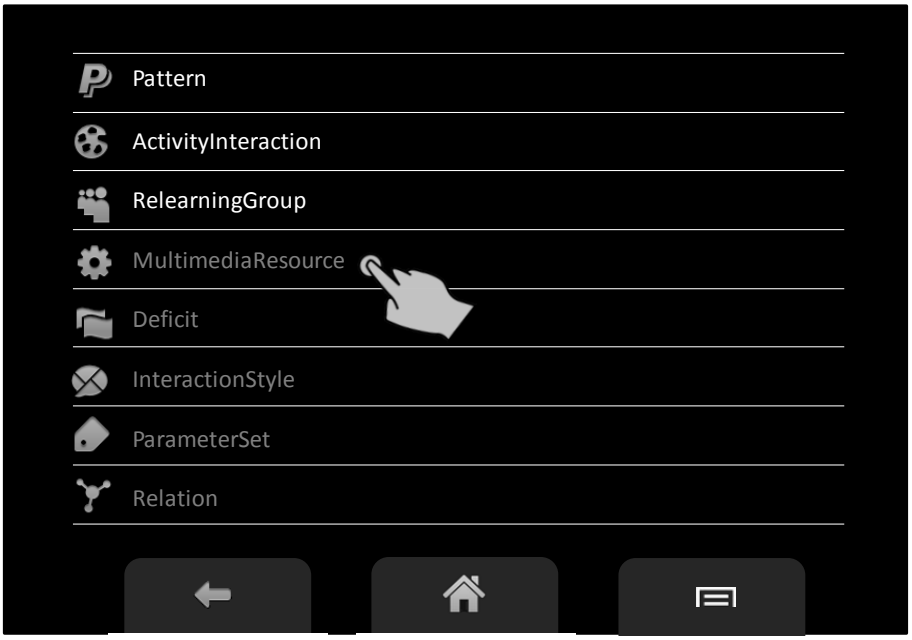

Fig. 6. Generated UIs and gestures for Heuristic #4

- *Heuristic #4* – [Association (1-directional) – Disable]. This association is used, for instance, to describe the unidirectional relationship between *activity interaction* and *multimedia resources* (see Fig. 3). As Fig. 6 shows, to specify a *multimedia resource* for a *relearning pattern*, an *activity interaction* must be previously defined. That is, unidirectional relationships describe dependencies in the order that instances from the metamodel can be created. *Disable* is the interaction used to reflect the dependencies created by unidirectional relationships, those elements having an incoming unidirectional relationship will be disabled in the user interfaces until the dependency is fulfilled. In the example of Fig. 6, once an activity interaction is defined, *multimedia resources* can be created. Once at least a

multimedia resource has been created a UI similar to that presented in Fig. 5 will be offered to the user to relate it to an activity interaction.

- *Heuristic #5* – [Generalization – Tap]. In the example, different kinds of ABI *deficits* are shown, such as cognitive, physic or emotional/behavioral. In the UI generated from the metamodel, the user should be able to navigate through the types of *deficits* and to specify special deficits for each *relearning pattern*. *Tap* is the gesture used to navigate the generalization hierarchies. In our example, the user navigates the *deficits* types of our metamodel. Once a particular *deficit* type is selected (see Fig. 7), the user can manipulate its instances.

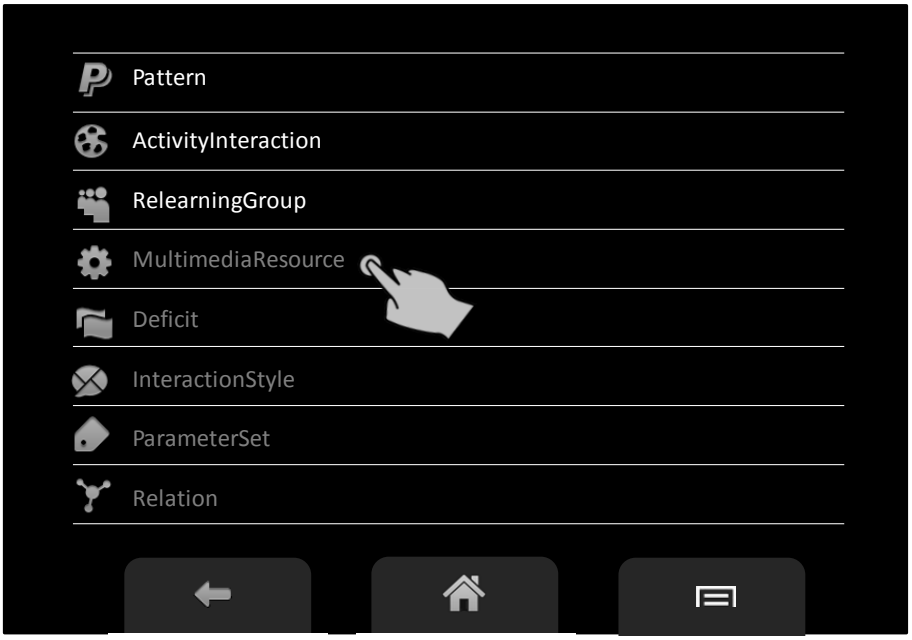

Fig. 7. Generated UIs and gestures for Heuristic #5

- *Heuristic #6* – [Composition – Press]. This relationship is used, for instance, in the ABI domain metamodel to specify that *model* is composed of 0 or many *ActivityInteraction*. This means that the user should be supported in creating, editing or deleting *activity interactions* instances. *Press* is the interaction style to be used for the manipulation of these relationships. A press is a touch in a surface for an extended period of time that is used to select the element to be modified, and then it can be drag to the bottom side of the UI for its edition or deletion. Fig. 8 shows how an *activity interaction* is manipulated. If a new instance has to be created, then the user presses on the free area of the UI so that a UI to specify the new instance is shown by the system. For instance, when the user presses on the free area of the Fig. 8 then the UI depicted in Fig. 9 is shown to specify a new *activity interaction*.

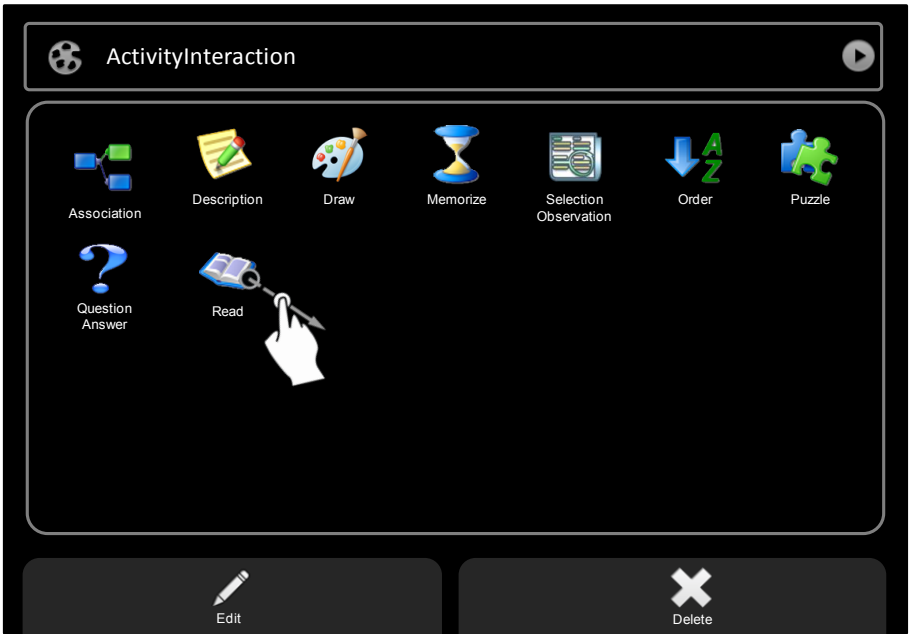

Fig. 8. Generated UIs and gestures for Heuristic #6 while editing or deleting an *activity interaction*

Fig. 9. Generated UIs and gestures for Heuristic #6 while creating an *activity interaction*

It is worth noting that although the ABI domain has been used as guiding example for the explanation of the proposed heuristics, they are totally independent of any domain. No information or relation to the domain has been considered while they were identified and automated by means of XPAND, so that they can be reused in different contexts by only feeding the generation tool with new metamodels. In addition, this proposal has a direct impact on productivity, as most MDD proposals do. This means that as the domain metamodel evolves, the multi-touch UI changes in an automatic as well without coding effort.

6 Exploratory evaluation

Both the heuristics and the metamodel infrastructure were validated and refined during the development of the HABITAT project by ABI specialists. This project aims at helping people with ABI. ABI patients have brain injuries caused by damage to the brain after birth that can involve the loss of cognitive, physical and/or emotional capabilities. The relearning process to improve their quality of life and help them in recovering their lost capabilities is a long and hard road.

This relearning process should be flexible since every single patient has specific needs to which the relearning process should be tailored as far as possible. This personalization should be achieved by using the specialists' normal vocabulary so that both the knowledge and the experience gathered during the relearning process can be reused. The issue of common vocabulary has already been addressed in previous studies [22].

The exploratory evaluation was conducted in collaboration with three ABI specialists. An iterative process was used for refining the heuristics on the basis of the feedback collected from the subjects. First, we met with the specialist to detect their needs in terms of modelling concepts. After several meetings, a metamodel was created that gathers the knowledge about the problem that is used to support the ABI specialists in creating their own relearning activity patterns to help during the relearning process of people with ABI. Second, a first version of the heuristics was defined, and they were used to create the first prototype that was tested by the ABI specialist. After several iterations the final version of the heuristics presented in this paper was produced. In each iteration, the cognitive walkthrough usability testing technique was used to detect possible usability issues in the prototype created by applying the heuristics.

Then, the engine to automate the generation of the user interface by applying the heuristics was developed. This engine was used to generate the multi-touch user interfaces for the ABI metamodel. This tool was used by the ABI specialists to instantiate the metamodel to create new relearning patterns.

The interviews with the ABI specialists revealed positive results, since all the specialists told that they were able to create the intended relearning pattern by using the generated user interface. Nevertheless, a thorough evaluation is required to fully validate all the heuristics, involving subjects from different domains.

7 Conclusions and further work

Model-Driven Development is a powerful tool to develop software. Nevertheless, some of the models involved in a model-driven development are actually dependent on a specific domain, where specific experts are actually the ones that should manipulate them. Nevertheless, usually domain experts are novice users. By providing a multi-touch based interaction, fully driven by the underlying metamodel, we are supporting novice users in the manipulation of metamodels. Our experience with ABI specialist have thrown successful results, supporting the specialist in creating relearning patterns [22] out of UI automatically derived from the domain metamodel.

In this paper a metamodels infrastructure and a set of heuristics are proposed. The heuristics drive the generation of multi-touch user interfaces that support the manipulation of metamodels to manipulate domain models. These heuristics exploit the semantics and structure of any metamodel specified by using *Ecore* to generate these multi-touch user interfaces.

The multi-touch user interfaces are generated so novice users can create models from metamodels, bridging the gap between the developer's language and what novice users understand. Furthermore, a metamodel infrastructure is also introduced to support the customization of the presentation of the different metamodel elements. Thus, icons or other aesthetics features can be specified to make the generated user interfaces more attractive to the user. Generating automatically a multi-touch user interface from metamodels also helps in improving certain basic usability principles, such as: visual presentation of task concepts, easy to learn, error prevention (by supporting the user in performing only valid actions), consistency (since the same tasks, such as adding items to an aggregation, is always carried out in the same way) or presentation structuring (by using the relationships in the metamodel to group related concepts).

As further work, we are applying these heuristics to other domains to gain further experience and improve our current tools for the generation of multi-touch user interfaces from domain metamodels.

Another future work is related to the evaluation of the proposal. Although the exploratory evaluation threw positive results, a more thorough evaluation is required to fully validate the heuristics presented. Currently several experiments are being designed, involving users from different domains.

Acknowledgements. We would like to thank the ADACE Association (http://www.adaceclm.org) for their cooperation in our research on ABI. This work has been partially supported by the grant insPIre (TIN2012-34003) from the Spanish Government Department of Science and Innovation and the grant AVANZA TSI-020400-2011-20 from the Ministry of Industry, Tourism and Commerce of Spain.

References

1. Apple. iOS Human Interface Guidelines (p. 156). Retrieved from http://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/MobileHIG.pdf, last access 05/02/2011.
2. Balzert, H. From OOA to GUI – The JANUS-System Human-Computer Interaction. IFIP TC13 Int. Conf. on Human-Computer Interaction (INTERACT '95), 27-29 June 1995, Lillehammer, Norway, pp. 319-324.
3. Balzert, H., Hofmann, F., Kruschinski, V., Niemann, C. The JANUS Application Development Environment - Generating More than the User Interface. 2nd Int. Workshop on Computer-Aided Design of User Interfaces (CADUI'96), June 5-7, 1996, Namur, Belgium, pp. 183-208.
4. Barclay, P. J., Griffiths, T., McKirdy, J., Paton, N. W., Cooper, R. and Kenne, J.. The Teallach Tool: Using Models for Flexible User Interface Design. 3rd Int. Conf. of Computer-Aided Design of User Interfaces (CADUI'99), October 21-23, 1999, Louvain-la-Neuve, Belgium, 139-158.
5. Bodart, F., Hennebert, A. M., Leheureux, J. M, and Vanderdonckt. Computer-Aided Window Identification in TRIDENT. IFIP TC13 Int. Conf. on Human-Computer Interaction (INTERACT '95), 27-29 June 1995, Lillehammer, Norway, pp. 331-336, 1995.
6. Booch, G., Rumbaugh, J., Jacobson, I. Unified Modeling Language User Guide, Addison Wesley, 1999.
7. Cachero, C. OO-H: Una expresión a los métodos OO para el modelado y generación automática de interfaces hipermediales. PhD. Alicante University, 2003.
8. Cicerone, K. D., Dahlberg, C., Kalmar, K., Langenbahn, D. M., Malec, J. M., Bergquist, T. F., Felicetti, T., Giacino, J. T., Harley, J. P., Harrington, D. E., Herzog, J., Kneipp, S., Laatsch, L., Morse, P. A. . Evidence-based cognitive relearning: Recommendations for clinical practice. Archives of Physical Medicine and Relearning, 81(12): 1596-1615, 2000.
9. Ciocca, G., Olivo,P., Schettini,R. Browsing museum image collections on a multi-touch table, Information Systems, 37(2): 169-182, 2012.
10. Consejería de Sanidad de la Junta de Comunidades de Castilla-La Mancha. Conjunto mínimo de datos básicos de la Región de Castilla La Mancha, 2006.
11. EMF, http://www.eclipse.org/modeling/emf/, last access 05/10/2011.
12. GMF. http://www.eclipse.org/modeling/gmf/, last access 05/10/2011.
13. Hailpern, B., & Tarr, P. Model-driven development: The good, the bad, and the ugly. IBM Systems Journal 45(3): 451-461, 2006.
14. Janssen, C., Weisbecker, A., and Ziegler, J. Generating User Interfaces from Data Models and Dialogue Net Specification. INTERACT '93 and CHI '93, 24-29 April 1993, Amsterdam, The Netherlands, pp. 418-423.
15. López Jaquero, V., Montero, F., Molina, J.P., González, P. Fernández Caballero, A. A Seamless Development Process of Adaptive User Interfaces Explicitly Based on Usability Properties. 9th Conf. on Engineering for Human-Computer Interaction. EHCI-DSVIS'2004, Hamburg, July 11-13, 2004.
16. Luebbe, A., Weske, M., Tangible Media in Process Modeling - A Controlled Experiment. 23rd Int. Conference Advanced Information Systems Engineering (CAISE 2011), London, UK, June 20-24, 2011, pp. 283-298.
17. Märtin, C. Software Life Cycle Automation for Interactive Applications: The AME Desing Environment. 2nd Int. Work. on Computer-Aided Design of User Interfaces (CADUI'96), June 5-7, 1996, Namur, Belgium, pp. 57-76.

18. Miller, J., & Mukerji, J. MDA Guide Version 1.0.1, Object Management Group, Needham, Massachusetts, June 2003.
19. Milisev, D. Model Driven Development with Executable UML, Wiley Publishing, Inc., Indianapolis, USA, 2009.
20. Molina, P.J., Meliá, S., Pastor, O. User Interface Conceptual Patterns, Interactive Systems. Design, Specification, and Verification. DSV-IS 2002, Rostock Germany, June 12-14, 2002, LNCS, Vol. 2545, Springer Verlag, Berlin, Germany. .
21. Montero, F., López Jaquero, V. IdealXML: An Interaction Design Tool and a Task-Based Approach to User Interface Design. 6th Int. Conference on Computer-Aided Design of User Interfaces (CADUI 2006), Bucharest, Romania, 6-8, June, 2006.
22. Montero, F., López-Jaquero, V., Navarro, E., & Sánchez, E. Computer-aided relearning activity patterns for people with acquired brain injury. Computers & Education, 57(1): 1149-1159, 2011.
23. Müller-Tomfelde, C. (ed.), Tabletops - Horizontal Interactive Displays, London: Springer London, 2010.
24. Navarro, E., López-Jaquero, V., Montero, F. HABITAT: a Web Supported Treatment for Adquired Brain Injured. IEEE Int. Conference in Advanced Learning Technologies (ICALT 2008), Santander, 1-5 July, 2008, pp. 464-466.
25. OMG, Meta-Object Facility (MOF), ptc/03-10-04 (MOF 2.0 Core Final Adopted Specification), http://www.omg.org/mof/, last visit 2011.
26. Pawson, R., Matthews, R. Naked Objects. Companion of the 17th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA '02). ACM, New York, NY, USA, 36-37. 2002.
27. Puerta, A. R., A model-based interface development environment. IEEE Software, 14 (4): 40-47, 1997.
28. Reichart, D., Forbrig, P., and Dittmar, A. Task models as basis for requirements engineering and software execution. 3rd Int. Work. on Task Models and Diagrams for User Interface Design (TAMODIA 2004), November 15 - 16, 2004, Prague, Czech Republic pp.51-58.
29. Roberts, D., Berry, D., Isensee, S., and Mullaly, J. Designing for the User with OVID: Bridging User Interface Design and Software Engineering. New Riders Publishing, 1998.
30. Schneiderman, B. Direct Manipulation for Comprehensible, Predictable and Controllable User Interfaces. In Proceedings of Intelligent User Interfaces, Orlando, USA, 1997.
31. Selic, B. The pragmatics of model-driven development. IEEE Software, 20(5): 19-25, 2003.
32. Toronto Acquired Brain Injury Network. http://www.abinetwork.ca/definition.htm, last access 02/05/2011.
33. XPAND, http://wiki.eclipse.org/Xpand, last access 02/05/2011.
34. Vanderdonckt, J. A MDA-Compliant Environment for Developing User Interfaces of Information Systems. CAiSE 2005, Porto, Portugal: 16-31, 2005.