Supporting Transformations across User Interface Descriptions at Various Abstraction Levels

Mauro Lisai, Fabio Paternò, Carmen Santoro, Lucio Spano

Supporting Transformations Across User Interface Descriptions at Various Abstraction Levels

Mauro Lisai , Fabio Paternò, Carmen Santoro, Lucio Davide Spano

CNR-ISTI, HIIS Laboratory,
Via Moruzzi 1, 56124 Pisa, Italy
{mauro.lisai, fabio.paterno, carmen.santoro, lucio.davide.spano}@isti.cnr.it

Abstract. Model-based approaches for user interfaces exploit various models in order to represent interactive systems at different levels of abstraction. During the design and development process, it is useful to have transformations to derive higher or lower level models. Such transformations should be customizable by designers to reach the desired results. In this paper we present a tool that allows designers without deep knowledge of transformation languages in creating and executing such transformations.

Keywords: Model-based Design, Model-to-model transformation

1 Introduction

Model-based approaches for User Interfaces (UI) have been applied in many domains, for example to address the problem of creating multi-platform UIs, in which abstract descriptions are transformed first into concrete descriptions and then into various implementation languages. In such approaches, model-to-model (M2M) and model-to-code (M2C) transformations play an important role, as their results should be as close as possible to the designer's expectations. Given that different transformation outputs are desirable for different use cases, the possibility to create and modify transformations is important. Different transformation languages have been proposed in the literature. We focus on those that have been exploited in particular for user interface descriptions. Since many UI specifications are XML-based, it is possible to exploit XSLT (eXtensible Stylesheet Language Transformations) for specifying the transformations. However, even if XSLTs are powerful and supported by many tools, they have a complex syntax that is not really suitable for many UI designer's background. Transformation Templates [1] have been proposed in order to parameterize the transformation logic, according to a set of parameter types. They are relevant for our work and we would like to obtain easier to manage UI transformations representations. In particular, we have considered MARIA [2] (Model-based lAnguage foR Interactive Applications), an XML-based set of model-based languages for which the proposed transformation tool can be applied to define mappings between abstract and concrete levels, as well as the transformations towards implementation languages based on XML.

2 Tool Support

In this section we describe the interactive environment supporting editing transformations that we propose. Figure 1 shows the UI for the definition of a transformation. Once this part has been activated, the designer can select the desired source and the target meta-model from a drop down list. Each meta-model is represented through a tree view, which allows the designer to easily recognize the hierarchical structure of the composing elements. Figure 1 shows the representation of MARIA desktop CUI (Concrete User Interface) as source and HTML5 as target metamodel (a transformation between these two languages has been specified with this tool). The tree nodes can be collapsed or expanded in order to hide unnecessary nodes, allowing the designer to focus only on the meta-model elements that are needed for defining the current transformation. When a tree node is selected, the corresponding meta-model entity attributes are displayed on the right panel (in Figure 1 the HTML5 body element attributes are visualized on the right panel).

A transformation rule can be defined first selecting an element from the source meta-model (that will be highlighted) and then selecting the destination element from the target meta-model. This action creates a simple transformation rule that maps the source to the target element without any particular condition. An arrow that connects the source and the target nodes represents the existence of such rule.

When the rule has been created, it is possible to add attribute mappings through a dedicated dialog. It shows the list of the currently defined mappings, allowing the designer to create new associations by selecting the source and the target attributes or to remove existing ones. This procedure is sufficient for defining *single* transformations. Moreover, *multiple* transformations can also be defined, iterating the process for creating a single one. Indeed, when more than one arrow starts from a source element, the tool prompts the designer to select the multiple transformation type (conditional, sequential or hierarchical). According to the selected type, the designer can add conditions, ordering and hierarchy values.

A *single* transformation rule can be used for instance in order to define a one-to-one mapping between a MARIA XML *Image* and a HTML *img*. The *sourceImage* attribute can be mapped to the *src* attribute of the *img* element.

A *conditional multiple* transformation rule can be defined adding conditions for the execution of each component. This can be done specifying, through a dedicated dialog, the *selection rule*. The dialog allows the designer to enter a list of conditions between two attributes, or between attributes and a static value. The condition operators that can be selected are different according to the selected attributes data type. An example of such transformation is the MARIA *grouping* mapping towards different HTML5 elements. According to its *role* attribute, a *grouping* can indeed represent one of the following HTML5 elements: *header, nav, section, article, aside* and *footer*. The multiple conditional rule checks the attribute value and generates the corresponding HTML element.

A *sequential multiple* transformation rule needs the specification of the ordering attribute. By default, when the designer selects this type of transformation, the ordering attribute is set according to the single component specification order (the arrow). However, the ordering attribute value is shown on the editor right panel, and it

is possible to change it directly. Such rule can be used for instance in order to map a MARIA *spin box* towards multiple HTML elements. Considering that this element is not included in HTML, the transformation maps it towards a button (with a plus label), a text field *input* element and another button (with a minus label).

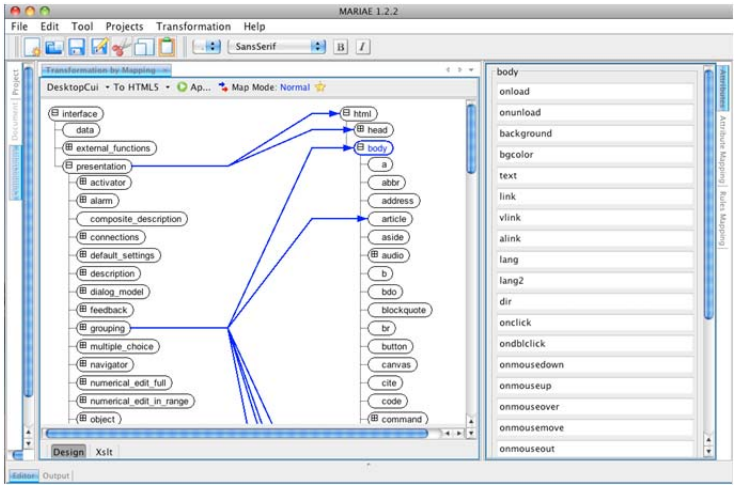

Fig. 1. The Transformation Editor User Interface.

For a *hierarchical multiple* transformation, the designer has to specify both the hierarchy and the ordering attribute. By default, the tool creates a new nesting for each component, increasing the current maximum hierarchy value. Default values can be changed through the editor right panel. Such transformation is useful especially in case of model refinement, as it happens in MARIA, where the CUI languages are obtained by adding refinements to the elements of the abstract language. For instance it is possible to refine a MARIA AUI (Abstract User Interface) activator into a desktop CUI button. In this case, multiple elements have to be generated, which should also be nested (the specification of the label is inside the *button* element, which is in turn contained into the *activator* element).

It is also possible to define transformations that map only attributes, without generating target elements. The procedure is the same as before, the designer specifies the different transformation type by pressing a button on the toolbar. Only attribute mappings are displayed using a different colour for the corresponding arrows: blue is used for mappings that generate target elements, red is used with only attribute mappings.

A first user test has been carried out in order to evaluate the effectiveness of the transformation meta-model and the usability of the tool. The test involved 11 users, all male, 27 years old on average. The participants had experience with UI development, and good knowledge of modelling techniques. However, all participants had very little knowledge of transformation languages. Thus, they represent our target users: UI designers without a deep knowledge of transformation techniques.

The participants were required to complete five tasks: loading the source and the target meta-models, editing a single transformation rule, editing a multiple transformation rule, editing an only-attribute mapping, and saving and loading a

transformation model. At the end, users were asked to complete a questionnaire, evaluating the transformation meta-model and the tool features on a 1 to 5 scale (with 1 the most negative value and 5 the most positive one). The main results are reported in terms of mean values and standard deviations.

The assessment of arrow-drawing paradigm for creating single transformations was on the positive side (4.45 + 0.69), although some participants expressed some complaints related to some confusion occurring when selecting target elements, due to a lack of knowledge of either the source or the target meta-model. One user proposed introducing a suggestion feature: when the designer is drawing an arrow, the nodes that are semantically similar with the source one should be highlighted with a different colour. Also tool support for creating multiple transformation rules was considered on the positive side (3.73 + 0.90). The participants suggested introducing a brief explanation of the three types of multiple transformations that can be defined with the tool. Moreover, users suggested that it should be possible to identify the type of multiple transformation directly from its graphical representation: the arrow should have a different colour or some other kind of indication. There should also be the possibility to see a summary of a multiple transformation, listing the values of the hierarchy and/or ordering attributes for each component at once. Overall, the evaluation provided encouraging feedback regarding the ways to represent transformations.

3 Conclusions and Acknowledgments

In this paper we discuss a tool for the specification of user interfaces transformations across specifications at various abstraction levels. The tool prototype allows UI designers to create their own transformations and to apply them to the UI meta-models. The tool has been integrated in a model-based authoring environment. The first evaluation of the tool provided positive feedback regarding the transformation meta-model with its tool support. Future work will be dedicated to investigating further refinements in the tool support, together with further empirical evaluation of the approach proposed.

We gratefully thank the support from the EU ICT SERENOA Project (http://www.serenoa-fp7.eu/).

References

1. Aquino, N. and Vanderdonckt, J. and Pastor O. Transformation templates: adding flexibility to model-driven engineering of user interfaces. In Proceedings of the 2010 ACM Symposium on Applied Computing (SAC'10), ACM, 1195--1202.
2. Paternò, F. and Santoro, C. and Spano, L. D. MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. ACM TOCHI, Volume 16, Issue 4, 2009, 19:1--19:30.