A Survey of Model Driven Engineering Tools for User Interface Design

Jorge-Luis Pérez-Medina, Sophie Dupuy-Chessa, Agnès Front
Laboratory of Informatics of Grenoble
385 rue de la bibliothèque, B.P. 53
38041 Grenoble Cedex 9, FRANCE
{Firstname.Lastname}@imag.fr

Abstract. The introduction of new technologies leads to a more and more complex interactive systems design. In order to describe the future interactive system, the human computer interaction (HCI) domain uses specific models and tools. In another way, the Model Driven Engineering (MDE) approach has been proposed in software engineering domain in order to provide techniques and tools for dealing with models in an automated way. MDE approach is based on models, meta-models, models transformation and models weaving and aims to produce productive models, i.e. models concentrated on their generative power. Considering these two domains and the already existing HCI works in MDE, the goal of this paper is to understand actual HCI design needs and to study how MDE tools can support HCI needs. As a first response, it proposes a survey of existing MDE tools in regards to HCI model management.

Keywords: HCI, MDE, model, meta-model, transformation, MDE tools, User Interface Design.

1 Introduction

Model-based approaches aim at helping developers understand user needs and design solutions in an effective way. In the HCI domain, models can be declarative in order to describe the future interactive system, but also generative to (semi-) automate the code generation. If the quality of the generated interfaces can be disappointing [22], models remain interesting for their declarative power. As a matter of fact, interactive systems are more and more complex: they can use everyday life objects to propose tangible interfaces; they can couple the virtual and the physical worlds in augmented reality systems; they can adapt themselves to the user context, etc. They are increasingly difficult to design. So new models appear to represent augmented reality systems [11, 27] or the user context (with a user model, a platform model and an environment model [28]).

In terms of tools, the HCI community uses different tools to support the design of interactive systems, e.g. CTTE [21], GUIDE-ME [32] K-MADe [4], and Teresa [5]. These tools mainly give support to model editing for task models (CTTE, Teresa and K-MADe) or specific models such as ASUR models (GUIDE-ME). In addition, some

of them [33, 4] allow model simulation. However, many others operations are possible on models, in particular to increase their generative power.

Model management aims at providing techniques and tools for dealing with models in more automated ways. It has been studied independently for years by several research communities in the context of databases, document management and software engineering. Nowadays, a federative approach emerges: model driven engineering (MDE [14]). At the origins of the movement, the Object Management Group proposes the Model Driven Architecture for object-oriented technologies. But this dependence on a technology and the absence of clear concept definitions lead to a more general approach, MDE. In MDE, any kind of models can be taken into account. So MDE is spreading quickly, in particular in the HCI domain as can be seen by the recurring workshop "Model Driven Development of Advanced User Interfaces" at one of the main conferences about MDE, MoDELS.

Based on related work on MDE for HCI, this paper tries to understand the HCI actual design needs related to MDE and proposes a survey of MDE tools for HCI. Our goal is not to identify the best tool for HCI design but to find criteria that could help HCI designers in the choice of a MDE tool.

The paper is organized as follows. Section 2 provides the basic definitions of MDE concepts. Section 3 describes the existing HCI works related to MDE. Section 4 provides a survey of MDE tools for HCI in terms of metamodeling, model transformation and others operations. Finally, conclusions are presented.

2 MDE concepts

2.1 Models and meta-models

MDE is a recent paradigm where code is not considered as the central element of software. Code is an element, a model produced by merging different modeling elements. So in MDE, everything can be considered a model. Minsky [20] defines that "To an observer B, an object M* is a model of an object M to the extent that B can use M* to answer questions that interest him about M". This definition shows a model is an object intended to represent a particular behavior, dependent on a particular disciplinary context. In the context of MDE, interesting models are those that can be formalized to make them productive. Some authors integrate this limitation directly into the definition of the notion of model: a model is a description of (part of) a system written in a well-defined language [18]. This definition makes an explicit reference to the notion of well-defined language. In MDE, such a language is described by a meta-model. A meta-model is a specification model that defines the language for expressing a model. It defines the concepts that can be used in the models, which conform to it. In this way, a meta-model allows designers to specify their own domain-specific languages. Models and meta-models are the first main concepts in MDE.

2.2 Model transformation

Another important concept in MDE is transformation. A transformation permits, from given models, to produce any model [19]. The model produced by transformations can be code, test cases, graphical modeling models, etc. The goal of transformations is double: on the one hand, they capitalize on know-how; on the other hand, they permit to automate this know-how. So transformations provide the generative power of models.

There are several kinds of generation. Classically, code can be generated from given models. But in reverse engineering, the models are produced from the code. There are many examples of translation of a model to another model such as the generation of UML models from formal specifications. In MDE, all these operations on models are considered as transformations. This is one of the key ideas in MDE that permits to consider all the generative operations in the same manner.

A difficulty remains in finding a language to express the transformations. Many different kinds of transformation languages exist: graphical languages like TrML[1]; XML XSLT-based[2] languages; languages based on a programming language (for instance, JMI[3] expresses Java-like transformations); ad-hoc languages like MOLA [17] and MTL [33]; and finally languages based on the OMG standard QVT[4]. QVT principles have been implemented in several languages, of which ATL (ATLAS Transformation Language [1]) that is currently most widely used.

2.3 Model Weaving

MDE is not limited to model transformations. [9] argues that transformations are not sufficient to manage the generative power of models and proposes another operation called model weaving. Model weaving [9, 10] is an operation on models that specifies different kinds of links between model elements. In order to explain model weaving, let us consider the simple information system for a library described in [10]. In this context, an example of transformation of one relational database R1 into its equivalent XML representation X1 is proposed (Figure 1). A model weaving operation is specified to capture the links between both schemas with all the information semantically relevant.

These links are represented in the R1_X1 mapping as illustrated in figure 1. In this example, both schemas represent the same information but distinct data structures are used. For instance, whereas the subjects have a *Name* in R1, they are called *Descr* in X1. The equality between these elements can be represented by the Equals links in the weaving. Moreover, one must also take into account the structure of both schemas: the foreign key constraints and the nested elements are respectively represented by *FK* and *Nested* links.

This example shows that a weaving is specific to a domain. The weaving relationships, e.g. “Equals” or “Nested”, depend on the concepts of the models to be

[1] TrML. Transformation modelling language, http://www2.lifl.fr/west/trml/
[2] W3C. World Wide Web Consortium, http://www.w3.org/TR/2007/REC-xslt20-20070123/
[3] JMI. Java Metadata Interface, http://java.sun.com/products/jmi/
[4] Query/View/Transformation. OMG Specification, http://www.omg.org/docs/ptc/05-11-01.pdf

manipulated. Thus, a weaving, like any model, must be in accordance with a meta-model. It allows afterward to define transformations from the mapping.

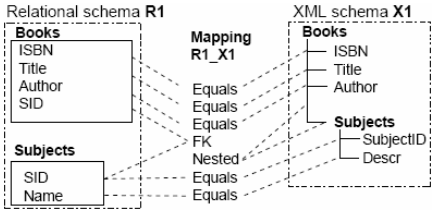

FIG. 1. Links between a relational and an XML schema of a library.

Model management is not limited to model transformation or weaving. Other kinds of operations can be applied to models. Models can be simulated, consistency can be checked between them, etc. If these operations are important to make models more useful, they are generally not presented as part of MDE for MDE concentrates on the generative power of models. We can note that it is important that MDE tools can be easily connected to other tools that will provide other operations on models.

3 Existing HCI works in MDE

Model-based Systems for User Interfaces Design (UIDE) have been addressed using many approaches over the years. Early works on UIDE such as Foley [15] established the foundations for transforming high-level specifications into executable code. Later, various approaches have been developed in the field of model-based design of interactive applications [24]. More recently, works in UI design are using partially the MDE principles. This section describes the existing works in order to identify needs related to MDE tools.

3.1 Models and meta-models in HCI

Historically, MDA and consequently MDE approaches have been "inspired" by concepts of the UML meta-model and the MOF meta-meta-model. MOF is a model of the meta-models proposed by the OMG. In particular, it is the meta-model of the most used meta-model, the UML one. MDE uses UML class diagrams as notation for the representation of models and meta-models.

In HCI, UML models are not widely used because they are not adequate but also because the HCI domain has developed its own notations such as task models, ASUR models, etc. Several meta-models have been proposed for context-adaptive user interfaces [28, 6, 7]. Generally, they include a meta-model for the task model, but also models related to the user context such as a platform model. For example, Fig. 2 represents a task meta-model proposed in [28]. In this meta-model, the tasks are

linked by operators. Logical and temporary operators are considered as binary, whereas the decorations on the tasks are supplied by unary operators.

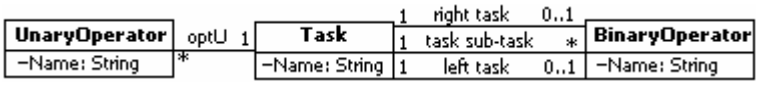

FIG. 2. A task meta-model [28].

The use of MDE and meta-models is not limited to the adaptation of the user interface to its context. Other domains of HCI also define meta-models for specific notations such as ASUR, a graphical notation for augmented reality systems [12] or for specific tools like in [16].

All these meta-models are independent, but they are instances of the same meta-meta-model (i.e. MOF). They are defined from scratch without being the extension of well-known meta-models. Another approach could be to extend an existing meta-model. In particular, UML proposes profiles to extend the UML meta-model to a specific domain. So the meta-models defined as UML profiles take advantage of the already existing semantics of UML and must conform to its semantics. For instance, some extensions have been proposed for HCI through UMLi [25] and for context-sensitive user interfaces [31].

The study of these existing works leads us to conclude that user interfaces design needs MDE tools, which support domain-specific meta-models and models. Unlike for software engineering (SE), there is no consensus on the models for HCI. In addition, even different notations are proposed for task modeling. So the HCI domain must manage several meta-models for task models. This diversity brings the need to use MDE tools that permit designers to create their own meta-model or to modify an existing one.

Finally if designers want to create links between HCI and SE models, all the meta-models must be instance of the same meta-model. As SE and MDE communities use the MOF as the meta-meta-model reference, it is important that the HCI domain conforms to this practice. So the HCI meta-models must be instance of the MOF and they must be represented by an UML class diagram.

3.2 Model Weaving in HCI

Establishing links between model elements can provide numerous application scenarios, such as model comparison, traceability, matching or interoperability. To our current knowledge, model weaving has been used in the HCI domain on the notion of mapping [29]. In this approach, a UI is described as a graph of models and mappings both at the design time and run-time.

The mappings are specified manually in a semi-formal way by the designer, or are created automatically by the system as the result of a transformation function. At design time, the mappings convey some properties that help the designer in selecting the most appropriate transformation function (e.g. the concepts manipulated within a task are grouped together). Either the target element of the mapping is generated

using a transformation function. At run-time, mappings are keys for reasoning on usability (e.g. select the appropriate usability framework in the generation of UIs). Mappings models are more than a simple traceability link; they can embed transformation in order to manage models consistency.

The use of model weaving is currently limited in HCI. It is more complex than the direct transformations or comparisons as it requires the creation of a weaving meta-model. But it increases the traceability of model manipulations by explicitly representing links between models. Then transformations or model comparisons can be more easily executed from the weaving links. So the need of weaving models in HCI is important.

3.3 Model transformations in HCI

More than weaving, transformation operations represent the heart of the MDE. Section 2.2 showed that there are several kinds of transformations and that many languages have been proposed to represent them. In this section, we study how the HCI community uses transformations for user interface design.

3.3.1 Transformation languages chosen in HCI

Many transformations languages are currently proposed and still developed in the MDE domain. An important decision consists in selecting a suitable language for transformations. Our study of existing works suggests that transformation languages are currently underused by the HCI community. Most of the work studied does not refer to any transformation language, which suggests that transformations are currently done in an ad-hoc manner or not formalized at all. Nevertheless, there are exceptions. In the domain of web interfaces, the transformation language is XSLT. In other domains, several papers [28, 7, 16] refer to ATL.

So it may be too early to clearly specify the HCI needs in terms of transformation languages. The HCI community seems to follow the standard of use. Nevertheless, the choice of a transformation language requires it to be easy to understand and to use, especially for non-MDE specialists as can be HCI designers. So it is important to note for each MDE tool which kind of language it supports.

3.3.2 Transformations proposed in HCI

In section 2.2, we identified the needs to generate code from models, models from code or models from models. Even if reverse engineering exists in HCI [3], we did not find any examples of model generation from code using MDE approach.

The idea of transforming one model into another is proposed mainly to bridge the gap between HCI and SE models. [23, 8] propose some informal transformations between activity diagrams and task model. But transformations are more commonly used to produce code. A good example of model transformation can be found in [29]. It describes a complete approach based on transformations with the generation of models from models and of code from models. Because of space limitations, we will comment only one transformation that generates one model from another. The rules are expressed in the same way to generate code.

Based on a case study of a Home Heating Control System (HHCS), this example shows that a final UI can be defined by a set of model transformations that follows the following steps: from the domain-dependent concepts and task models, an abstract UI (Workspace) is derived; this abstract UI is then transformed into a concrete UI (CUI), which is transformed into the final UI. To give a more precise example, we shall concentrate on the transformation from tasks into workspaces. In this example, the tasks are transformed into workspaces; the operators between tasks into chains between workspaces.

Figure 3 presents the meta-models used in the transformation of the tasks into workspaces. In this figure, we see that every task is associated with a workspace and that the binary operator gives rise to chains between workspaces.

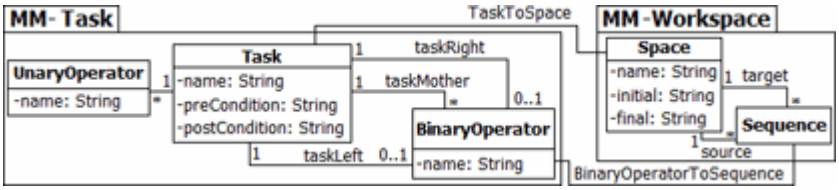

FIG. 3. Meta-models used in the transformation from task to workspace [29].

In the current implementation of HHCS, the mappings between the task model, the workspace and the CUI are expressed in ATL; an example is illustrated in figure 4. The first rule illustrates the generation of a task into a workspace; it consists in creating a space for every task with the assignment of the name of the task. The second rule illustrates the transformation of a binary operator into a chain; it considers only the operator "Or" and is written in two parts: the first one consists in the selection of the binary operators of type "or"; the second describes the access given by the space representing the mother task to spaces representing their two daughters.

```
Module M2A-T to M2A-E {
--Rule 1: The Space takes the name of the task
rule TaskToSpace {from t: M2-T!Task to e: M2-E!Space ( name <- t.name )}
--Rule 2: Management of the operators of type "or"
rule OperatorOrToSequence { from o: M2-T!BinaryOperator ( o.name ="or" )
sequenceLeft : M2-E!Sequence (
    source <- [Task2Space.e]o.taskMother,
    target <- [Task2Space.e]o.taskRight,),
to sequenceRight : M2-E!Sequence (
    source <- [Task2Space.e]o.taskMother,
    target <- [Task2Space.e]o.taskLeft,), }
```

FIG. 4. Example of the transformation Task to Workspace in ATL [28]

In MDE, there is no distinction between transformations: a transformation always generates one model from another. It is assumed that the code or program is also a model. Nevertheless, in the perspective of using MDE tools, one important aspect is to guarantee that the transformation result can be expressed in a recoverable format that is useful for another tool. This implies that the format of the transformation result is important. It is needed to know if the result is a text file that can be compiled or

interpreted or if it is a structured file (in XML for instance) that can be manipulated by design tools.

In the perspective of comparing MDE tools according to HCI needs, we note that the existing works in HCI reflects a clear need to realize transformations of HCI models. To go further, the HCI community could define libraries of classic transformations that could be integrated and manipulated by MDE tools. So it is important that MDE tools propose a transformation repository or at least the load of existing transformations. This brings the need to a common language to express transformations but also this adds constraints on the format to permit interoperability between tools. We note that the format of the transformation result is also important to know in order to determine the future operations that can be realized on the resulting model.

4 Survey of MDE tools for HCI

4.1 Diversity of Tools

Both at the commercial and research levels, several tools for MDE are either available or in development. These tools are designed as frameworks [2] or as plug-in [1]. Several classification works [13, 26] and tool comparisons [30] were proposed. However, no classification estimates the functional criteria that we defined towards our needs, in particular in terms of specific models used in HCI domain.

Table 1 shows a list of tools that we have considered realizing our survey. This list is focused on the MDE tools which could be used in the HCI domain as the manipulated models are not limited to UML models.

Table 1. Survey of MDE Tools.

Tool	Version	Description
ACCELEO GPL - Open source	2.0.0	Eclipse and EMF template-based system for MDA generation. http://www.acceleo.org/pages/accueil/fr
AndroMDA Open source	3.2	An extensible generator framework. Models from UML tools will be transformed into deployable components for your favorite platform (J2EE, Spring, .NET). http://galaxy.andromda.org/index.php?option=com_frontpage &Itemid=48
ADT Open source	2.0	ATL Development Tools are a suite of Eclipse plugins including an ATL engine (compiler and virtual machine) as well as an IDE. http://www.sciences.univ-nantes.fr/lina/atl/atldemo/adt
AToM3 Open source	2.2	A Tool for Multi-formalism and Meta-Modelling supporting modelling of complex systems. http://atom3.cs.mcgill.ca/index_html
DSL Tools (Visual Studio 2005 SDK)	4.0	DSL Tools enable the construction of custom graphical designers and the generation of source code using domain-specific diagrammatic notations in Visual Studio 2005. http://msdn2.microsoft.com/en-us/vstudio/aa718368.aspx
Kermeta	0.4.1	A metamodeling language which allows describing both the structure and the behaviour of models. http://www.kermeta.org/
ModFact GPL - Open source	1.0.1	A tool that provides a framework for building application. http://modfact.lip6.fr/
Merlin Open source	0.5.0	A software modelling tool based on model transformation and code generation. http://merlingenerator.sourceforge.net/merlin/index.php
MDA Workbench Open source	3.0	The MDA Workbench is a MDA tool implemented as an Eclipse plug-in based on modelling and code generation. http://sourceforge.net/projects/mda-workbench

MOFLON Open source	1.1.0	A meta modelling framework built as plug-in for the graph transformation tool Fujaba. http://www.moflon.org/
OptimalJ Professional Edition	3.0	Generator of J2EE applications using patterns to translate business models into working applications. http://www.compuware.com/ products/optimalj/
QVT Partners BSD like license	0.1	Tools based on QVT for transformation models to models and code generator. http://qvtp.org/downloads/qvtp-eclipse/
SmartQVT Open source	0.1.4	A model transformation tool based on QVT-Operational language. http://smartqvt.elibel.tm.fr/
UMLX Open source	0.0.2	An experimental concrete syntax for a transformation language. http://dev.eclipse.org/viewcvs/indextech.cgi/gmt-home/subprojects/UMLX/

These tools will be studied according to the needs listed in the previous sections. These needs are general to the HCI domain. Any HCI designer must refine them to choose his MDE tool. So we do not intend to find the best tool but rather to provide relevant information to choose a MDE tool. We will present our survey in terms of the MDE important concepts: models and meta-models, operations on models and other functionalities.

4.2 Tools in terms of meta-models and models expression

Regarding models and meta-models, the HCI community needs tools that do not just consider UML models, but also specific models. Our list of tools being limited to this kind of tools, any tool in the list can be suitable for HCI in terms of model and meta-model support. Nevertheless, to refine our comparison, we introduce a criterion about the way of expressing models and meta-models: models and meta-models can be represented either textually or graphically. We also note if constraints can be added to complete models and meta-models. Constraints are written in OCL, the constraint language for UML.

Table 2. MDE tools in terms of meta-models and models expression.

Tools	Expression (Meta-models)		Expression (Models)	
	Graphical (G) or Textual (T)	Constraints	Graphical (G) or Textual (T)	Constraints
ACCELEO	G, T	OCL	G, T	OCL
AndroMDA	T	OCL	G, T	OCL
ADT	T	OCL	T	OCL
AToM3	G	-	G	-
DSL tools	G, T	-	G, T	-
Kermeta	G,T	OCL	G, T	OCL
ModFact	G	-	G	-
Merlin	G,T	OCL	G, T	OCL
MDA Workbench	G, T	OCL	G, T	OCL
MOFLON	G, T	OCL	G, T	OCL
OptimalJ	G	OCL	G	OCL
QVT Partners	G, T	OCL	T	OCL
SmartQVT	T	OCL	T	OCL
UMLX	G, T	OCL	G, T	OCL

From the previous table, we would recommend that a user interface designer should better choose a tool allowing a graphical expression of models and meta-

models, because graphical representations are of course easier to use than textual representations for non specialists.

4.3 Tools in terms of model transformation and weaving

As mentioned in section 3, HCI needs in terms of operations on models are not limited to transformations. Table 3 lists all the model manipulations proposed by the tools and shows that only ADT provides some part of the infrastructure for the manual creation of weaving models, what is a real advantage on other tools.

Then for transformations, even if there is a standard specification for transformations (QVT), there is no standard language. The majority of MDE tools support QVT so that, in principle, the use of QVT guarantees that the result of a transformation is compatible with another tool that uses QVT. But in practise, the implementations of QVT are different and the compatibility between tools is not guaranteed. We also showed in section 3.3 that XSLT and ATL were nowadays the only two languages used by the HCI community. So to support the creation of transformations libraries for HCI, the tools ADT and UMLX, which support XSLT and ATL, should be preferred in the HCI domain. Moreover ATL is already widely used in the SE domain. So ATL is a good candidate to facilitate links between HCI and SE models.

Moreover it is important to identify the form (text or model) of the generated models in order to identify which kind of tools can manipulate them. In table 3, the word "Text" is used when the result of a transformation is textual. Generally the result is some code written in a programming language (java, C, C++, Cobol, Fortran, VB.net, etc.) that can be compiled or interpreted. The term XMI is used when the result of the transformation is a model in the XMI form (XML Metadata Interchange), which can be loaded in many design tools. Here again ATL and UMLX (with other tools) have an advantage as they provide the XMI and the textual format.

Considering model operations, two tools are good candidates for the HCI domain: ATL that is the solution for works in the SE spirit and UMLX which is more adapted for works with web technologies.

Table 3. MDE tools in terms of models transformation and weaving.

Tool	Transformation				Weaving
	Language	Graphical (G) or Textual (T) Expression	Generated model		
			XMI	Text	
ACCELEO	QVT, JMI	T	-	Yes	-
AndroMDA	ATL, MofScript	T	Yes	Yes	-
ADT	ATL	T	Yes	Yes	Yes
AToM3	Multi formalism (python)	G		Yes	-
DSL tools	Notation XML	T	Yes	Yes	-
Kermeta	QVT	T	-	Yes	-
ModFact	QVT	T	-	Yes	-
Merlin	QVT, JET	T	-	Yes	-
MDA Workbench	QVT	T	-	Yes	-
MOFLON	JMI	G	-	Yes	-
OptimalJ	QVT	T	-	Yes	-
QVT Partners	QVT	T	Yes	Yes	-

SmartQVT	QVT	T	Yes	Yes	-
UMLX	XSLT, QVT	T	Yes	Yes	-

4.4 Tools in terms of other operations

The studied MDE tools offer good solutions for meta-modeling and transformations. But one may want to reuse models, meta-models or transformations into another tool, so it is very important to know the capacity of a tool to interoperate with other tools.

In sections 3.1 and 3.3, we noted the importance of the format to exchange models and meta-models and to bridge the gap with the SE domain. A great part of the tools is centred on the MOF specification. So they can cover the modelling needs of different domains and especially of HCI. Several implemented formats have been proposed for the MOF: ECore, MDR (Metadata Repository), KM3 (Kernel Meta-Meta Model), DSL (Domain Specific Language) and CWM (Common Warehouse Meta-model). Nevertheless, DSL does not conform to MOF's implementation. That's why KM3 was created: KM3 is a specialized language to specify meta-models and is used as a bridge between MOF and DSL. The most used format is ECore, which is a simplified version of the MOF. Moreover MDE tools provide many libraries of predefined models and meta-models in ECore. So the choice of a ECore compliant tool is important to guarantee the development and the exchange of reusable models and meta-models.

Regarding model transformation, XMI is proposed for transformations but it is not so widely chosen. As a matter of fact, many other tools prefer textual transformations, in particular for QVT tools. In terms of interoperability, Eclipse proposes de facto methods for the storage and the recovery of models based on XMI. So the great majority of MDE tools is based on Eclipse and can interoperate with other Eclipse tools.

Finally, what is more important in the HCI domain is the interoperability of MDE tools with existing HCI design tools. Generally HCI design tools do not have a known meta-model. However the models produced with them can be saved in an XML format. The interoperability between MDE and HCI design tools can be easily guaranteed by transforming every XML file in a ECore compatible format, so that it could be recovered by the MDE tools that support this format. A longer term solution is that HCI tools incorporate the MDE standards and create mechanisms to import or export information based on the XMI format.

Table 4. MDE tools in terms of other operations.

Tool	Repository			Interoperability with others tools
	Metamodeling	Model transformation	Constraints	
ACCELEO	DSL, MDR, ECORE	-	XMI	Eclipse, Netbeans
AndroMDA	MOF, DSL	-	XMI	Eclipse
ADT	DSL, KM3, MDR, ECORE	Text (ATL)	XMI	Eclipse, Netbeans
AToM3	Proprietary graphical multi - formalism			-
DSL tools	DSL - Proprietary notation	XML / XMI	-	Eclipse, Netbeans
Kermeta	ECORE	Text (QVT)	XMI	Eclipse
ModFact	ECORE	XMI	XMI	Eclipse

Merlin	ECORE	Text (QVT)	XMI	Eclipse
MDA Workbench	ECORE	XMI	XMI	Eclipse
MOFLON	ECORE	-	XMI	Eclipse
OptimalJ	CWM, ECORE	XMI	XMI	Eclipse
QVT Partners	ECORE	Text (QVT)	XMI	Eclipse
SmartQVT	ECORE	Text (QVT)	XMI	Eclipse
UMLX	ECORE	XMI, XSLT	XMI, XSLT	Eclipse

5 Conclusion

The goal of this paper is to propose a survey of MDE tools in order to help the HCI community in the choice of a MDE tool. Considering existing works in the HCI domain, we think that the HCI domain shows a clear need for the MDE approach and tools. First, considering models and meta-models, HCI designers use a lot of domain-specific models such as task models, ASUR models, etc. that conform to specific meta-models. Transformation models and weaving models are also needed in HCI domain. In particular, model weaving has been used on the notion of mapping where a user interface is described as a graph of models and mappings both at design time at run-time. Moreover, transformations allow to generate code from models, but also to produce new models from other ones. Two types of transformations are then needed, those that generate code (more generally, a text file that can be compiled or interpreted) and those that generate graphical models (more generally, a structured file that can be manipulated by design tools).

Based on these needs, we draw a survey of several MDE existing tools. Several conclusions can be drawn from this comparison. In terms of modeling, a great part of the tools are centered on MOF and allow to model domain-specific models. In terms of transformations, there is no standard language to use, but it is important to know the language manipulated by the tools and to specify if they are graphical or textual. Moreover, it is important to know the format (text or model) of the generated models in order to identify the kind of tools that can then manipulate them. Our conclusion is that MDE is able to answer the specific needs of the HCI community in terms of models. Nevertheless, the HCI community has to incorporate the proposed standards that MDE is nowadays using. We hope this comparison will be useful to any HCI designer who wants to select a MDE tool based on functional needs in terms of graphical (or textual) expression of domain specific models, models transformation, models weaving and interoperability with specific HCI tools.

Acknowledgments. We would like to express our special thanks to Stéphanie Marsal-Layat for her help in the tool survey. We are also grateful to the Federation IMAG, the Foundation “Gran Mariscal de Ayacucho, the university UCLA-Venezuela” for their financial support.

References

1. Allilaire, F., Idrissi T.: ADT: Eclipse development tools for ATL. In Proceedings of the 2nd European Workshop on Model Driven Architecture (MDA) with an emphasis on Methodologies and Transformations (EWMDA-2). Computing Laboratory, University of Kent, Canterbury, UK. England. September (2004), 171-178.
2. Amelunxen C., Königs A., Rötschke T., Schürr A.: MOFLON: A Standard-Compliant Metamodeling Framework with Graph Transformations. In: Rensink A., Warmer J., (eds), Model Driven Architecture – Foundations and Applications: 2nd European Conference, Heidelberg: Springer Verlag, 2006; LNCS, Vol. 4066, pp. 361-375, (2006).
3. Bandelloni R., Paternò F., Santoro C.: Reverse Engineering Cross-Modal User Interfaces for Ubiquitous Environments. In Proceedings of the Engineering Interactive Systems Conference (EIS'07). Salamanca, Spain 2007; (to appear in LNCS). Springer Verlag.
4. Baron M., Lucquiaud V., Autard D., Scapin D.: K-MADe : un environement pour le noyau du modèle de description de l'activité. In Proceedings of 18th French-speaking conference on Human-Computer Interaction (IHM'06), ACM Press. Montréal, (2006), 287-288.
5. Berti S, Correani F., Mori G., Paternò F., Santoro C.: TERESA: A Transformation-Based Environment for Designing Multi-Device Interactive Applications. In Proceedings of CHI 2004, CHI '04 extended abstracts on Human factors in Computing Systems, ACM Press. Vienna, Austria. April (2004), 793-794.
6. Boedcher A., Mukasa K., Zuehlke D.: Capturing Common and Variable Design Aspects for Ubiquitous Computing with MB-UID. In: Proceedings of the International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI'05) organized at MoDELS'05, Jamaica, October (2005). CEUR Workshop Proceedings vol. 159 (2005).
7. Botterweck G.: A Model-Driven Approach to the Engineering of Multiple User Interfaces. In Proceedings of the International workshop on Model Driven Development of Advanced User Interfaces (MDDAUI'06) organized at MoDELS'06. Genoa, Italy, October 2006; LNCS, vol. 4364, pp. 106-115, Springer (2007).
8. Brüning, J., Dittmar A., Forbrig P., Reichart D.: Getting SW Engineers on board: Task Modelling with Activity Diagrams. In Proceedings of the Engineering Interactive Systems Conference (EIS'07). Salamanca, Spain 2007; (to appear in LNCS). Springer Verlag.
9. Didonet Del Fabro M., Bézivin J., Jouault F., Breton E., Gueltas G.: AMW: a generic model weaver. In Proceedings of the 1ère Journée sur l'Ingénierie Dirigée par les Modèles (IDM'05). Paris, France, June 2005. Sébastien Gérard, Jean-Marie Favre, Pierre-Alain Muller, Xavier Blanc, Editors Paris, Paris, France (2005), 105-114.
10. Didonet Del Fabro, M., Jouault, F.: Model Transformation and Weaving in the AMMA Platform, In Proceedings of the International Summer School in Generative and Transformational Techniques in Software Engineering (GTTSE'05). Braga, Portugal, July 2005; LNCS, Vol. 4143, pp. 71-77, Springer Verlag, (2007).
11. Dubois, E., P.D., G., Nigay, L.: ASUR++: a Design Notation for Mobile Mixed Systems. Interacting With Computers, Vol. 15 (2003), 497-520.
12. Dupuy-Chessa S., Dubois E.: Requirements and Impacts of Model Driven Engineering on Mixed Systems Design. In Proceedings of the 1ère Journée sur l'Ingénierie Dirigée par les Modèles (IDM'05). Paris, France, June 2005. Sébastien Gérard, Jean-Marie Favre, Pierre-Alain Muller, Xavier Blanc, Editors Paris, Paris, France (2005), 43-54.
13. Eclipse Modeling Project. Official site: http://www.eclipse.org/modeling/, February (2007).
14. Favre J.-M.: Towards a basic theory to model driven engineering. 3er UML Workshop in Software Model Engineering (WISME'04) joint event with UML'04. Lisboa, Portugal, October (2004). Available online at http://www.metamodel.com/wisme-2004/papers.html
15. Foley J., Sukaviriya N.: History, Results, and Bibliography of the User Interface Design Environment (UIDE), an Early Model-based for User Interface Design and Development.

In Paternò F. (ed;) Interactive Systems: Design, Specification, Verification, Springer Verlag, (1994), 3-14.
16. Ian Bull R., Favre J.M.: Visualization in the Context of Model Driven Engineering, In: Proceedings of the International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI'05) organized at MoDELS'05, Jamaica, October (2005).
17. Kalnins A., Barzdins J., Celms E.: Model Transformation Language MOLA. In Proceedings of Model-Driven Architecture: Foundations and Applications (MDAFA'04). Linkoeping, Sweden, June 10-11, (2004) 14-28.
18. Kleppe A., Warmer S., Bast W.: MDA explained: The model-driven architecture: Practice and promise; Addison-Wesley, 192 pages. April (2003).
19. Mens T., Van Gorp P.: A Taxonomy of Model Transformation. Electronic Notes in Theorical Computer Science, Vol. 152. March (2006), 125-142.
20. Minsky M.: Matter, Minds, and Models. In Proceedings of International Federation of Information Processing Congress (Vol. 1), New York, United States. (1965), 45-49.
21. Mori G., Paternò F., Santoro C.: CTTE: Support for Developing and Analyzing Task Models for Interactive Systems Design, IEEE Transactions on Software Engineering, August 2002 (Vol. 28, No. 8), IEEE Press, 797-813.
22. Myers B., Hudson S.E., Pausch R.: Past, Present, and Future of User Interface Software Tools, ACM Transactions on Computer-Human Interaction, Vol 7, Issue 1 (2000), 3-28.
23. Nóbrega L., Jardim N., Coelho H.: Mapping ConcurTaskTrees into UML 2.0, In: Stephen W. Gilroy, Michael D. Harrison (Eds.): Interactive Systems, Design, Specification, and Verification: 12th International Workshop on Design, Specification, and Verification of Interactive System (DSVIS'05), July 13–15, (2005), Newcastle upon Tyne, England, (2005); LNCS 3941, Springer 2006, 237-248.
24. Paternò F.: Model-Based Design and Evaluation of Interactive Application. Springer Verlag, (1999).
25. Pinheiro da Silva P., Paton N.: User Interface Modeling in UML*i*. IEEE Software, Vol. 20 No. 4, July/August (2003), 62-69.
26. Planet MDE, Official site: http://planet-mde.org/index.php?option=com_xcombuilder&cat=Tool&Itemid=47, Sept, (2007).
27. Shaer O., Leland N., Calvillo E.H., Jacob, R.J.K.: The TAC Paradigm: Specifying Tangible User Interfaces. In Personal and Ubiquitous Computing, vol. 8, no. 5 (2004) 359-369.
28. Sottet J-S., Calvary G., Favre J-M., Coutaz J., Demeure, A., Balme, L.: Towards Model Driven Engineering of Plastic User Interfaces. In Satellite Proceedings of the ACM/IEEE 8th International Conference on Models Driven Engineering Languages and Systems, MoDELS/UML 2005, Springer LNCS, (2005), 191-200.
29. Sottet, J-S., Calvary, G., Coutaz, J., Favre, J.-M.: A Model-Driven Engineering Approach for the Usability of Plastic User Interfaces. In Proc. of the Engineering Interactive Systems Conference (EIS'07). Salamanca, Spain 2007; (to appear in LNCS). Springer Verlag.
30. Tariq N., Akhter N.: « Comparison of Model Driven Architecture (MDA) based tools » Karolinska University Hospital; A Thesis Document, Sockholm, Sweden. June (2005), 74p.
31. Van den Bergh J., Coninx K.: Using UML2.0 and Profiles for Modeling Context-Sensitive User Interfaces, In: Proceedings of the International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI'05) organized at MoDELS'05, Jamaica, October (2005). CEUR Workshop Proceedings vol. 159 (2005).
32. Viala, J. Dubois, E., Gray, P.: GUIDE-ME: Environement Graphique de Manipulation de la Notation ASUR. In: ACM Proceedings of the French conference: Mobilite et Ubiquite. 2004. In: Canals, G., Giboin, A., Nigay, L., Pinna, A.-M., Tigli, J.-Y. (eds), Nice, France, June (2004), 74-78.
33. Vojtisek D., Jzquel. J.-M,: MTL and Umlaut NG: Engine and Framework for Model Transformation. ERCIM News, Nro. 58, Special Issue on Automated Software Engineering, July (2004), 42-45.