A Tool For Generating Model Transformations By-example In Multi-agent Systems

Ivan Garcia-Magarino, Sylvain Rougemaille, Rubén Fuentes-Fernández, Frédéric Migeon, Marie-Pierre Gleizes

A Tool for Generating Model Transformations By-Example in Multi-Agent Systems

Iván García-Magariño[1], Sylvain Rougemaille[2], Rubén Fuentes Fernández[1], Frédéric Migeon[2], Marie-Pierre Gleizes[2], and Jorge Gómez-Sanz[1]

[1] D. Software Engineering and Artificial intelligence
Facultad de Informatica - Univesidad Complutense Madrid, Spain
{ivan_gmg,ruben}@fdi.ucm.es, jjgomez@sip.ucm.es
[2] Institut de Recherche en Informatique de Toulouse
SMAC Research Group
Université de Toulouse, France
firstname.name@irit.fr

Abstract. Many Multi-Agent Systems (MAS) methodologies incorporate a model-driven approach. Model Driven Engineering is based on three main ideas: models are the "first-class citizens", meta-models define modelling languages that are used to specify models and models are transformed during the development. However, model transformation is still a challenging issue in MAS. At first, MAS designers are not necessarily familiar with existing model transformation languages or tools. Secondly, existing tools for creating model transformations do not satisfy the necessities of agent-oriented software engineering, since they focused on coding with little support for developers. This paper proposes a tool for the creation of model transformations that is based on the generation of model transformations by-example. This tool overcomes the limitations of other similar tools in the sense that it can generate many-to-many transformation rules. The tool application is exemplified with two MAS methodologies, *INGENIAS* and *ADELFE*.

1 Introduction

Developing *Multi-Agent Systems* (MAS) in the scope of modern information systems is a demanding activity. MAS are usually related to distributed applications in changing environments, where knowledge is partial and there are requirements of a flexible behaviour. Different proposals have been made to alleviate the designer's work, mainly through methodologies and their support tools. This work gathers the experience of two different research groups that developed the agent-oriented methodologies *INGENIAS* [10] and *ADELFE* [1]. These two methodologies use Model Driven Engineering (MDE) principles to carry out the development process. One of the backbones of these approaches are model transformations which allow the automation of tasks and code generation. *ADELFE* proposes model-to-model transformations to integrate several modelling languages and separate concerns into different models [15]. *INGENIAS* also raises the need of model-to-model transformations for designer assistance and model refinement.

The specification of these transformations is still a critical point since only MDE specialists are able to do it. In this paper, we propose a further step in the application of MDE in the scope of MAS development. Model Transformation By Example (MTBE)

proposes to generate model-to-model transformations from representative source and target models so that, transformation specification task would no more be assigned to specialists. However, existing tools adopting this approach do not satisfy yet the requirements of model transformations in MAS. The work presented here describes a tool for the creation of model transformations based on MTBE. It overcomes the limitations of other similar tools since it can generate many-to-many transformation rules.

This paper introduces some practical uses of MTBE for model refactoring purpose, firstly in the scope of *INGENIAS*, for which was originally design the MTBE tool, and secondly to the *ADELFE* methodology. The *INGENIAS* methodology [11] is devoted to the development of multi-agent systems, it was based on the use of meta-modelling techniques. It covers the whole development cycle, from analysis to implementation and provides tool support with the *INGENIAS Development Kit (IDK)* [5], which follows the principles of *Model-Driven Development*. *INGENIAS* and the IDK have been applied successfully in several areas; for instance, in surveillance [12], mobile tourist guide [9] and social simulation [8]. *ADELFE*[1] [1] is an agent-oriented methodology for designing Adaptive Multi-Agent System (AMAS) [4]. It proposes specific modelling languages such as *AMAS-ML* (AMAS Modelling Language) and integrates a model-driven implementation phase based on model transformations [14].

From the experience acquired during the definition of these two methodologies, we can state that MDE are specially appropriate to increase the productivity of MAS development. MTBE and the proposed tool can strengthen the whole development process, because it provides means for MAS designers to define their own model transformations.

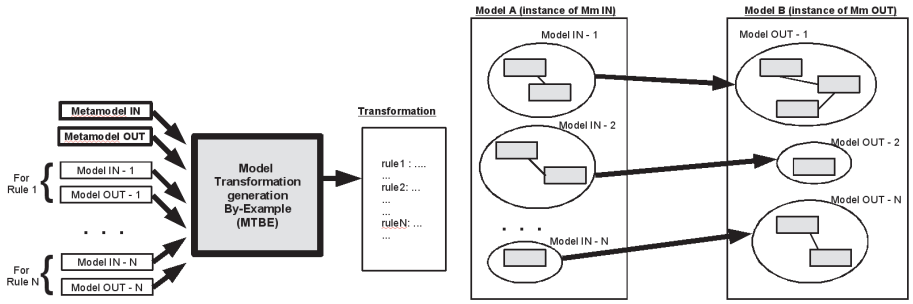
The paper focuses on the application of MTBE in the scope of MAS. The next section depicts the principles and steps of the model transformations generation process. Section 2 introduces the interests of the proposed MTBE tool as well as its implementation. Section 4 presents two practical model refinement applications of MTBE, these examples are taken from both *INGENIAS* and *ADELFE* methodologies. In Section 5 we present some other works that deal with MTBE as well as MAS methodologies that could take advantage from it. Finally, the last section presents the conclusions we can draw from these experiments and proposes some specific further issues that have to be coped with in the application of MDE to MAS development.

2 Principles of Model Transformation by Example

As a recent initiative, Model transformation By-Example (MTBE) [16] is defined as the automatic generation of transformations from source and target model pairs. The common steps of MTBE are the following:

1. *Manual set-up of prototype mapping models*. The transformation designer assembles an initial set of interrelated source and target model pairs.
2. *Automated derivation of rules*. Based upon the available prototype mapping models, the transformation framework should synthesise (see Figure 1(a)) the set of model transformation rules. These rules must correctly transform (see Figure 1(b)) at least the prototypical source models into their target equivalents.

[1] ADELFE is a French acronym for "Atelier de Développement de Logiciels à Fonctionnalité Emergente".



(a) The inputs and outputs of MTBE (b) The Behavior of the Generated Transformation

Fig. 1. Description of the Model Transformation By-Example (MTBE)

3. *Manual refinement of rules*. The transformation designer can refine the rules manually at any time. However, MTBE recommends these modifications to be included in the pairs of models, so the alterations are not overwritten the next time the transformation is generated.
4. *Automated execution of transformation rules*. The transformation designer validates the correctness of the synthesised rules by executing them on additional source-target model pairs as test cases.

The MTBE approach avoids the hard-coding of transformations, which frequently hinders the principles of MDE. MTBE follows MDE principles because its main products are models and transformations. In addition, transformation designers in MTBE do not need to learn a new model transformation language; instead they only use the concepts of the source and target modelling languages.

3 Tool for Model Transformation by Example in MAS

Existing MTBE algorithms and tools [18, 16, 17] are only able to generate one-to-one transformations. For this reason, Grasia research group has defined an algorithm for MTBE that overcomes this limitation. This algorithm can generate many-to-many transformation rules. The input patterns of the many-to-many rules are simulated with constraints. The elements of the output pattern are defined directly with the transformation language and the connection among these elements are recursively defined by the algorithm. In addition, it can create the appropriate mapping of attributes so the generated transformation propagates the information from the source to the target. This *matching-control mechanism* relies on the use of identifiers in the source model that are referred in the target model. A prototype tool (see Figure 2) implements this algorithm for generating ATL (Atlas Transformation Language) transformations [7] from *INGENIAS* model; this tool is called *MTGenerator*.

The tool provides an interface (GUI) in which the user can select the input and output meta-models of the transformation. The user must define the meta-models with the ECore language and select the corresponding location paths in the top-left area of the

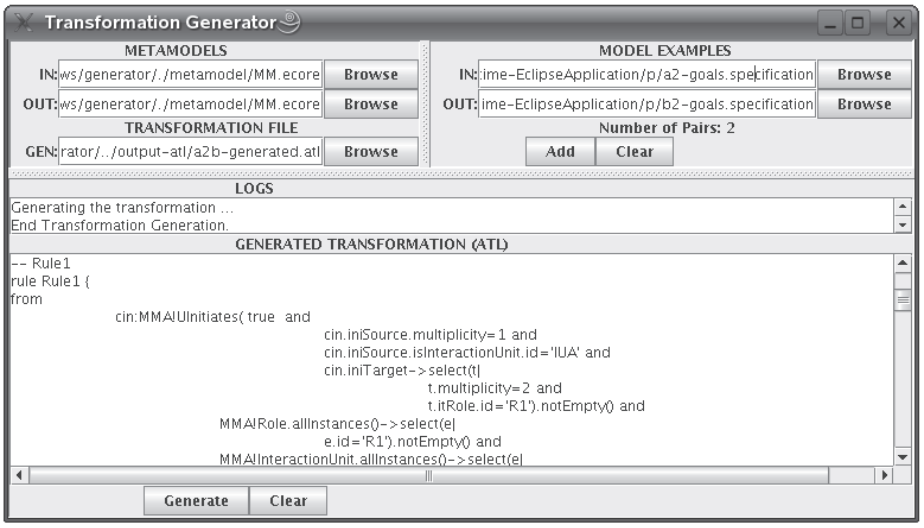


Fig. 2. Model-Transformation Generator Tool

GUI. The user can add the pairs of model with the top-right area of the generator tool, by selecting the corresponding location paths and adding them. After the automatic generation, the tool shows some logs in the *Logs* text area, confirming that the generation has finished successfully. The generated model transformation is shown in the bottom text area of Figure 2. In this manner, the user can examine the generated transformation. In brief, the presented MTGenerator tool automatically generates a model transformation. Even if the user wants to manually improve the generated transformation, the tool saves time for the user because it provides a generated transformation as a basis for the final model transformation.

4 Application of MTBE in MAS

MAS meta-models usually involves concepts semantically rich that have to be specified in terms of several meta-classes. Models conforming to MAS meta-models contain many instances of these meta-classes, as a consequence, their refinement or translation involve complex patterns of modelling elements. Therefore, MAS model driven development can profit considerably of many-to-many model transformation generation.

Furthermore, MAS meta-models used to be less stable than others, the concepts they define are still evolving as no consensus has been reached in the agent community. This implies that models need to be updated each time their meta-models undergo modifications. MTBE can help this upgrading task for models which meta-models have been modified. Source models are the models to upgrade, thus the only task left is the description of target models. The evolution of MAS concepts implies that this upgrade process is potentially more frequent in MAS methodologies. MTBE

constitutes a powerful means to reduce this upgrade time, as it allows MAS designers to generate the required transformations without having to assimilate transformation languages.

One application of MTBE in MAS is the automation of repetitive mandatory tasks which are often related to phase transition in the development process, as for instance the translation of requirements models in the beginning of *ADELFE* analysis phase. MTBE can also be used to improve the quality of models by defining specific automatic refactoring that prevent designers from potential mistakes. As MTBE helps designers to define their own transformations, they could embody easily their pragmatic knowledge of the application domain and improve the process. The following sections presents two examples of MTBE practical uses in the *INGENIAS* and *ADELFE* methodologies.

4.1 MTBE in INGENIAS

In *INGENIAS*, the MTBE is applied to create model transformations for assisting the designers in the creation of the model specification. There are several processes for modelling a whole MAS with *INGENIAS*. Most of the processes start with the definition of the use cases. For this reason, this paper presents the generation of a model transformation that creates the definition of roles from the specification of the use cases. In *INGENIAS*, the use case diagrams usually include roles and goals, which must be linked afterwards for defining the roles. In addition, at least, an agent must be created for playing each role. In this example we propose to automatically create the role and agent definitions with a model transformations.

In particular, Figure 3 shows the pairs of models, from which the model transformation was generated. In the first pair, for each role in the source model, the target model

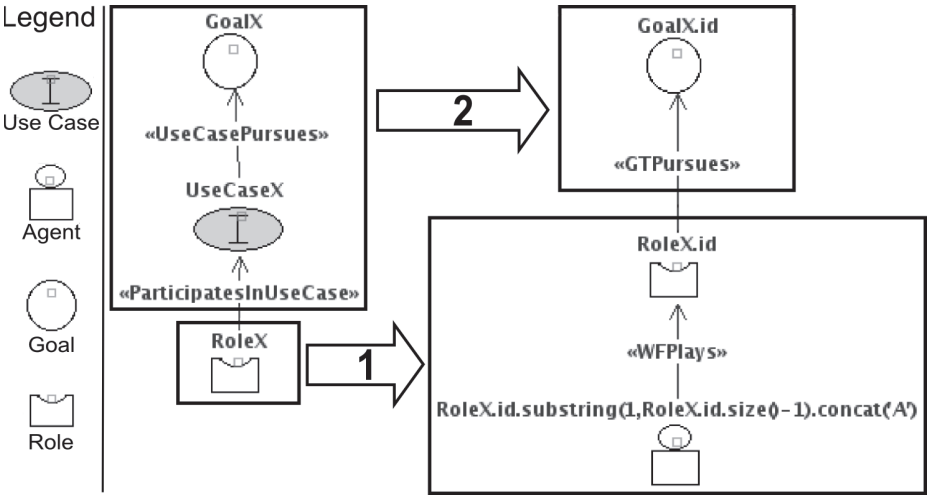


Fig. 3. Model transformations for generating the Specification of Roles. Each square represent a model example for MTBE. Each pair of models is related with an arrow and a number, in which the source and target models are respectively situated at the left and right sides of the arrow.

contains the role and an agent playing this role. The identifier of the role is copied whereas the identifier of the agent is an alteration of the role identifier. The expression of the agent identifier in the target model is defined according to the matching-control mechanism provided by our tool (see Section 2). In the second pair, the source model contains a goal connected to a role through a use case. The target model has the same goal that is directly linked with the role. In this manner, the model transformation link roles with the goals according to the existent use cases.

As one can observe, this example needs to transform patterns of several modelling elements; thus, one-to-one transformation rules do not satisfy the requirements of this transformation.

4.2 MTBE Use in ADELFE

One task of the *ADELFE* methodology design phase is the specification of agent interactions (direct communication between agents). The UML 2.0 sequence diagram is used to achieve this task. A model-to-model transformation has been specified to translate UML sequences of messages into AMAS-ML *Cooperative Interaction Protocols*. However, once the protocols are integrated to the AMAS-ML model messages emitting and reception have to be declared in the agents involved in these protocols. This example proposes to assist the designer by automating this process. The model-to-model transformation is created applying the MTBE principles (see Section 2).

Figure 4 shows the way the MTBE is specified via a meta-object notation (instances of AMAS-ML meta-classes). The source example model is figured on the left hand side. It presents a message (*m1*) that is owned by a protocol (*protocol1*) and sent by the *agent1* to the *agent2*. On the right hand side the figure presents the wished result of the transformation. The idea is to add a communication action (*cAm1*) to the *agent1* action

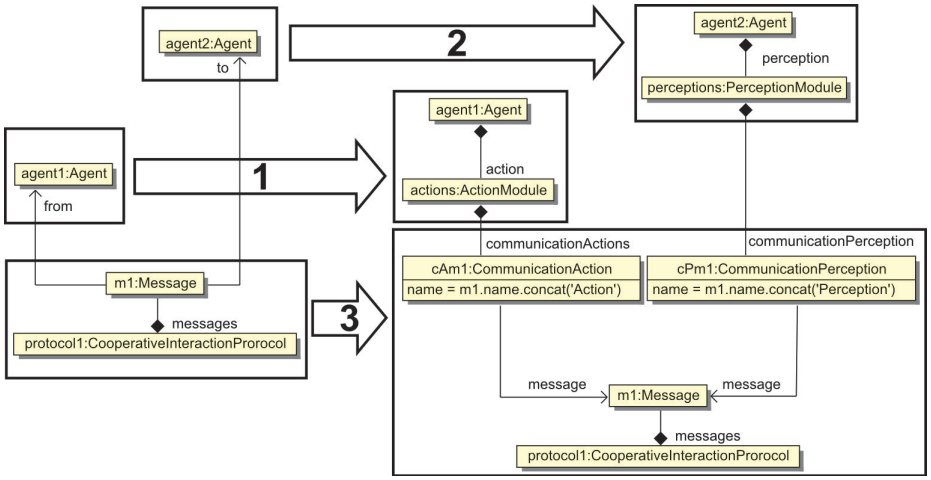


Fig. 4. Model refactoring example in ADELFE using MTBE : creating communication action and perception in respective modules for each message specified in AMAS-ML interaction protocols

module (*actions*) and the respective communication perception (*cPm1*) to the *agent2* perception module (*perceptions*). Both the created communication action and perception are related to the *m1* message (*cAm1.message* and *cPm1.message*) in conformance to the AMAS-ML meta-model. Furthermore, each numbered arrows represents a transformation rule. As a matter of fact, the result of the generation will be separated in three different parts (ATL rules). One (*arrow number 1*) to add the action module (*actions*) to the source agent (*agent1*), another (*number 2*) to do the same with the perception module (*perceptions*) of the target agent (*agent2*) and a last one (*number 3*) to integrate the communication action (*cAm1*) and perception (*cPm1*) to their respective modules and links them to the message (*m1*).

Although this process could be achieve by AMAS designers, we advocate that their productivity should be improved by a transformation that abstains them to perform these quite repetitive actions. Considering the set of protocols and messages that are usually defined in AMAS-ML models, the automation of this process could save a precious time during the design phase. In addition, this transformation could strengthen the design phase by avoiding errors while treating each messages from each protocols by hand.

5 Related Work

First of all, there are other MTBE tools. For instance, Wimmer et al.[18] present another MTBE tool which uses the same model transformation language: ATL. However, Wimmer et al. generate simple ATL rules that transforms only one isolated element into another one. Moreover, Varro and Balogh[16, 17] use inductive logic programming to derive the transformation rules with a MTBE approach. An innovation of this work is the learning of negative constraints from *negative examples*. A *negative examples* is a context of elements for which a rule do not have to be applied. In addition, this work carries what Varro and Balogh call *connective analysis*. In the connective analysis, the references among modelling elements are analysed. However this analysis is only successfully executed in the rule outputs. Their approach only generates one-to-one rules.

The great advantage of the work presented here over Wimmer's, Varro and Balogh's is the generation of many-to-many rules, by means of OCL constraints within the input side of the rules. In other words, the rules generated by the tool presented in Section 2 allow one to transform patterns of modelling elements into other patterns of modelling elements.

The most relevant features of the existing tools and the one we have presented are compared in Table 5

Besides *INGENIAS* and *ADELFE*, there are other MAS methodologies that use refinement model transformations such as Tropos[2] [3]. It is associated with a design tool called TAOM4E (Tool for visual Agent Oriented Modelling for the Eclipse platform) [2]. TAOM4E uses a specific modelling language and introduces a model driven approach. Perini et al. [13] presents the different types of transformation which were

[2] `http://www.troposproject.org/`

Table 1. Comparison of existent MTBE tools with the presented tool

Features of MTBE	Varro and Balogh	Wimmer et al	Our Technique
Mapping of attributes	yes	yes	yes
Propagation of links	yes	yes	yes
Negative Examples	yes	no	no
Generation of Constraints	no	yes	yes
Explicit Allocation of Target Elements	yes	no	yes
Limit number of input elements of rules	1	1	no-limit
Limit number of output elements of rules	1	1	no-limit

implemented in Tropos (model refinement and translation of preliminary UML2.0 models). In the same scope, *MDAD* (Model Driven Agent Development) applies MDE principles for the development of MAS [6]. It defines UML based meta-models (profile), covering aspects such as the domain, the agents and the organisation. In *MDAD* the preliminary abstract model is transformed into a platform specific model conforming to the INteractive Agent Framework (INAF) meta-model.

The Tropos and MDAD approaches as well as all MAS methodologies using models, can benefit from the presented MTBE tool. The MTGenerator tool (see section 2) provides means to assist MAS designers in defining many-to-many model transformation rules with an user friendly GUI. This kind of transformations is especially useful in MAS development.

6 Conclusions and Future Work

This paper presents a tool based on the MTBE principles. This tool can generate model transformations that satisfy the fundamental requirements in the scope of MAS. MTBE facilitates the task of the MAS designers and reduces design time by providing transformations generation. Moreover, MTBE speeds up the transformation specification as the process just consists of defining two models and using the tool to generate the transformation that relates them. MTBE brings a real improvement specially considering refining transformations (same source and target meta-models). MAS designers can easily defines new refactoring or refinement transformations as they are used to manipulate the concepts from both source and target models (they are MAS concepts). For the experimentation, this work includes two examples from two agent oriented methodologies: *INGENIAS* and *ADELFE*.

This particular aspect can be exploited to enhance development processes. Each transition from task to task or from phase to phase that needs specific treatments could be automated. Designers practical knowledge can be embodied into transformations that they are able to define thanks to MTBE. We foresee to automate most of the the *INGENIAS* and *ADELFE* process workflow definition with the help of the presented MTBE tool.

Future work with the generator tool includes that the user can define model-to-model transformations for a wider range of meta-models. Another future direction is to apply

the presented tool for exogenous model-to-model transformations (different source and target meta-models). Finally, the use of negative examples can be incorporated in the tool to facilitate the expression of complex constraints over source model elements.

References

1. Bernon, C., Camps, V., Gleizes, M.P., Picard, G.: Engineering Adaptive Multi-Agent Systems: The ADELFE Methodology. In: Henderson-Sellers, B., Giorgini, P. (eds.) Agent-Oriented Methodologies, pp. 172–202. Idea Group Pub., NY (2005)
2. Bertolini, D., Delpero, L., Mylopoulos, J., Novikau, A., Orler, A., Penserini, L., Perini, A., Susi, A., Tomasi, B.: A tropos model-driven development environment. In: Boudjlida, N., Cheng, D., Guelfi, N. (eds.) CAiSE Forum. CEUR Workshop Proceedings, vol. 231. CEUR-WS.org. (2006)
3. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems 8(3), 203–236 (2004)
4. Capera, D., Georgé, J.P., Gleizes, M.P., Glize, P.: The AMAS Theory for Complex Problem Solving Based on Self-organizing Cooperative Agents. In: TAPOCS 2003 at WETICE 2003, Linz, Austria. IEEE CS, Los Alamitos (2003)
5. Gómez-Sanz, J.J., Fuentes, R., Pavón, J., García-Magariño, I.: INGENIAS development kit: a visual multi-agent system development environment. In: AAMAS (Demos), pp. 1675–1676. IFAAMAS (2008)
6. Jarraya, T., Guessoum, Z.: Towards a model driven process for multi-agent system. In: Burkhard, H.-D., Lindemann, G., Verbrugge, R., Varga, L.Z. (eds.) CEEMAS 2007. LNCS, vol. 4696, pp. 256–265. Springer, Heidelberg (2007)
7. Jouault, F., Kurtev, I.: Transforming Models with ATL. In: Bruel, J.-M. (ed.) MoDELS 2005. LNCS, vol. 3844, pp. 128–138. Springer, Heidelberg (2006)
8. Pavon, J., Arroyo, M., Hassan, S., Sansores, C.: Agent-based modelling and simulation for the analysis of social patterns. Pattern Recognition Letters 29(8), 1039–1048 (2008)
9. Pavón, J., Corchado, J., Gómez-Sanz, J., Ossa, L.: Mobile Tourist Guide Services with Software Agents. LNCS, pp. 322–330. Springer, Heidelberg (2004)
10. Pavón, J., Gómez-Sanz, J.: Agent Oriented Software Engineering with INGENIAS. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS, vol. 2691, pp. 394–403. Springer, Heidelberg (2003)
11. Pavón, J., Gómez-Sanz, J., Fuentes, R.: Model Driven Development of Multi-Agent Systems. In: Rensink, A., Warmer, J. (eds.) ECMDA-FA 2006. LNCS, vol. 4066, pp. 284–298. Springer, Heidelberg (2006)
12. Pavón, J., Gómez-Sanz, J.J., Fernández-Caballero, A., Valencia-Jiménez, J.J.: Development of intelligent multisensor surveillance systems with agents. Robotics and Autonomous Systems 55(12), 892–903 (2007)
13. Perini, A., Susi, A.: Automating model transformations in agent-oriented modelling. In: Müller, J.P., Zambonelli, F. (eds.) AOSE 2005. LNCS, vol. 3950, pp. 167–178. Springer, Heidelberg (2006)
14. Rougemaille, S., Arcangeli, J.P., Gleizes, M.P., Migeon, F.: ADELFE Design, AMAS-ML in Action. In: International Workshop on Engineering Societies in the Agents World (ESAW), Saint-Etienne, mai 2008, pp. 213–224. Springer, Heidelberg (2008), http://www.springerlink.com/
15. Rougemaille, S., Migeon, F., Maurel, C., Gleizes, M.P.: Conception d'applications adaptatives basées sur l'IDM (Accepted). In: Artikis, A., O'Hare, G.M.P., Stathis, K., Vouros, G. (eds.) ESAW 2007. LNCS, vol. 4995, pp. 318–332. Springer, Heidelberg (2008)

16. Varro, D.: Model transformation by example. In: Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (eds.) MoDELS 2006. LNCS, vol. 4199, pp. 410–424. Springer, Heidelberg (2006)
17. Varró, D., Balogh, Z.: Automating model transformation by example using inductive logic programming. In: Proceedings of the 2007 ACM symposium on Applied computing, pp. 978–984 (2007)
18. Wimmer, M., Strommer, M., Kargl, H., Kramler, G.: Towards Model Transformation By-Example. In: Proceedings of the 40th Annual Hawaii International Conference on System Sciences, vol. 40(10), p. 4770 (2007)