MDA-based Interoperability Establishment Using Language Independent Information Models

Carlos Agostinho[1], Jaroslav Černý [1], Ricardo Jardim-Goncalves[1,2]

[1] Centre of Technology and Systems, CTS, Uninova, 2829-516 Caparica, Portugal
[2] Departamento de Engenharia Electrotécnica, Faculdade de Ciências e Tecnologia, FCT, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
{ca,rg}@uninova.pt

Abstract. Nowadays, more and more enterprises realize that one important step to success in their business is to create new and innovative products. Many times the solution to do that is to abandon the idea of an enterprise as an "isolated island", and get collaboration with others: worldwide non-hierarchical networks are characterized by collaboration and non-centralized decision making. This paper proposes a conceptual model common to the entire business network, in a framework that enables the abstraction of individual models at their meta-level and increase language independency and interoperability, keeping all the enterprise software's integrity intact. The strategy presented allows an incremental mapping construction, to achieve growing integration.

Keywords: MDA, MDE, Enterprise Interoperability, Model-Morphisms, Model and Data Transformations, Language Independent Information Models

1 Introduction

Interoperability is a property directly related with the heterogeneity of model languages, communication capabilities, databases and semantics. Differences in these hide a great barrier to achieve the time-to-market symbiosis that can unleash a solution more valuable than the sum of its creators. Interoperability is more than just a communication support: it is a software approach to maximize the benefits of diversity, rather than to integrate the different system into one. Such diversity leads to more fruitful results than by just integrating different systems into one. Since many organizations developed and purchased software solutions based on their own needs, the required cooperation with others is not a trivial activity and business partnerships are less effective, evidencing low level of interoperability.

To solve this problem, instead of adopting a paradigm that obligates every organization to migrate their systems, or develop complex mappings in a single step to comply with these advanced practices, one can act at the communication module, where the data is exchanged. The authors propose Model Driven Architecture (MDA) based technologies for the development of transformations and execution of automatic and executable Model Morphisms (MoMo), also providing traceability and repeatability on them. The proposed framework enables to respond automatically to

the network dynamics and its sustainability, i.e. changes that occur over the time and impact negatively the interoperable state can be tuned and balanced.

2 Model Driven Engineering

Model-Driven Engineering (MDE), sometimes also referred as Model-Driven Development (MDD), is an emerging practice for developing model driven applications. It represents a promising software engineering approach to address systems complexity, both by simplifying and formalizing the various activities and tasks that comprise an information system life cycle (i.e. from design, to construction, deployment, operation, maintenance and modification). Given today's increase of technology complexity, models are becoming a powerful mechanism to precisely describe problems in a way that avoids delving into technological details, thus allowing developers to focus on more abstract tasks and increasing productivity rather than to computing concepts. MDE is meant to maximize compatibility between systems, simplifying the process of design, and promoting communication between individuals and teams working on the system [1].

MDD/MDE's vision goes even further, invoking the unification principle, which states that *"everything is a model"* (i.e., platforms, components, legacy software, services, etc.), encouraging the support of models at different levels of abstraction, from high-level business models focusing on goals, roles and responsibilities down to detailed use-case and scenario models for business execution [2, 3]. These models are developed through extensive communication among product managers, designers, and members of the development team, and as they approach completion, enable a fast development of product and systems. However, despite obvious potential capabilities for closely matching the EI holistic levels, yielding major productivity and reliability benefits, there is not yet consensus about its technology readiness [4, 5].

2.1 Model-Driven Architecture (MDA)

Among the several realizations of the MDE/MDD principles that exist, such as Agile Model Driven Development [4, 6], Domain-oriented Programming [7], Microsoft's Software Factories [8] and Model Driven Architecture [9, 10], MDA is perhaps the most prevalent at the moment. Since it was launched, in 2001, MDA has been having a major impact on the software development community, and presently, there is a large landscape of tools available for its support.

MDA has as its foundation on three complementary ideas: direct representation, automation and open standards. The first makes use of abstract models to represent ideas and concepts of the problem domain, reducing the semantic gap existing between domain-specific concepts and the technologies used to implement them. The second uses model transformation tools to automate the translation process from high level specifications and formal descriptions of the systems, to the bottom levels and implementation code, therefore increasing speed, code optimization and avoiding human errors in the process. Regarding the last foundation, MDA enforces the usage of open standards to specify the high level models, and the features of the target

implementation platforms, promoting interoperability among the entire ecosystem of tool vendors [3, 11].

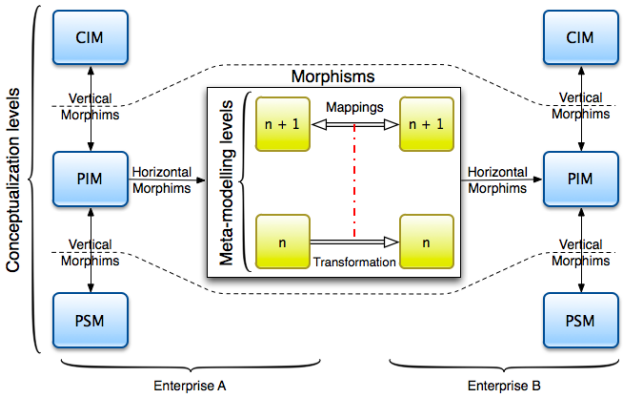

Fig. 1. MDA's Conceptualization Levels and Transformation Types (adopted from [12])

An MDA system can be observed and analyzed from different points of view (see Fig. 1), and in order to support the supra-cited foundations it defines a hierarchy of models at three different levels of information abstraction [13]: (i) Computation Independent Model (CIM), which specifies the requirements for the system and the environment where it will operate. It's also called a domain model since it's meant for the domain practitioners and it's based on the vocabulary of the specific target domain; (ii) Platform Independent Model (PIM), which is the formal specification of the structure and functionality of the system that abstracts away technical details. More concretely, it focuses on the operation details while hiding specific details of any particular platform in order to be suitable for use with several different platforms; (iii) Platform Specific Model (PSM) that combines the specification in the PIM model with details that specify how the system uses a particular type of platform. Thus, a PSM adds to the PIM, technical details and implementation constructs that are available in a specific implementation platform, including middleware, operating systems and programming languages (e.g. Java, C++, EJB, XML, Web Services, etc).

2.2 Model Transformations

Based on model transformations, the MDA unifies every step of the development of an application or integrated suite from its start as a CIM of the application's business requirements through PIM defined functions and behavior, one or more PSMs, to generated code and a deployable application. The PIM remains stable as technology evolves, extending and thereby maximizing software return on investment. Portability and interoperability are built into the MDA architecture, which also introduces the distinction between vertical and horizontal transformations (evidenced in Fig. 1).

Vertical Transformations: Imply a change on the abstraction level of the resulting model, e.g. going from PSM to PIM implies a generalization transformation, and from PIM to PSM implies a specialization transformation. The amount of generated code depends on both the code generator and also the level of detail represented in the PSMs (i.e. how well the PSM captures the details of the physical platform). Ideally, only small portions of missing code should have to be added by the human developer in order to ensure that the generated code and auxiliary files are ready for compilation, linking and deployment.

However, today, MDA vertical transformations are still an open issue. Incomplete applications have been developed from CIM to PIM due to the lack of efficient tools and methods for transformation [14]. For this purpose, new concepts, methods and tools are necessary.

Horizontal Transformations: In this case (e.g. refactoring of individual models, language translation, or even joining different models), the level of abstraction remains unchanged, leading to solutions for interoperability problems at the same enterprise level [15]. Both input and output models must be an instance of a well-defined meta-model, and have to be classifiable according to the meta-modelling level they belong to. Due to that, greater interoperability benefits but also harder complications are expected in horizontal transformations, since at the time of the transformation specification (mapping), one has to be concerned with different language-related specificities [12]. In fact, different languages might enable to describe the same objects with different detail levels (e.g. properties, constraints, etc.).

With horizontal transformations, companies can specify P2P mappings to translate any data from one format to the other, thus allowing an exchange of information. When performing this type of transformation (e.g. converting instances of a model to instances of another model) an explicit or an implicit mapping of the meta-model has to be performed. Thus, as depicted in Fig. 1, the idea is that when performing a transformation at a certain level "n", this transformation has (implicitly or explicitly) to be designed by taking into account mappings at level "n+1". Once the "n+1" level mapping is complete, executable languages can be used to implement the transformation, e.g. ATL[1] and the QVT[2]. This is valid either for CIM, PIM or PSM models.

Horizontal transformations, which are targeted in this paper's research, are traditionally static processes that once defined can be repeated any number of times achieving the same results. The major difficulty is defining them while supporting network dynamicity, joining the efforts of business and technical specialists at reduced costs.

3 Model Morphims (MoMo)

The concept of morphism is described in mathematics as an abstraction of a structure-preserving map between two mathematical structures [16]. Recently, this concept is gaining some meaning in computer science, more exactly in systems interoperability.

[1] ATL – Atlas Transformation Language (www.eclipse.org/m2m/atl/)
[2] QVT – Query View Transformation (www.omg.org/spec/QVT/)

This new meaning of Morphism describes the relations (e.g. mapping, merging, transformation, etc.) between two or more information specifications as the ones needed to define MDA horizontal transformations.

In this context, the research community identifies two core classes of MoMo: non-altering and model altering morphisms [15]. In the first, given two models (source *A* and target *B*), a mapping is created relating each element of the source with a correspondent element in the target, leaving both models intact. In model altering morphisms, the source model is transformed using a function that applies a mapping to the source model and outputs the target model. Other relations, such as the merge operation, can also be classified as model altering morphisms, however they are not detailed in this paper.

3.1 MoMo Formalization

The research community has developed many proposals to morphisms formalization [15]. Graph theory has been used in some, although other theories can be considered to achieve the envisaged goals, e.g., set theory [17], model management [18], or semantic matching [19]. However there is not a single perfect solution that can be used to achieve all the morphisms goals at once. Some are ideal for structural issues, others for semantics providing good human traceability, and others are more formal and mathematical based. Agostinho et al. ([20]) proposes a 5-tuple mapping expression, with the objective to consolidate and complement existent approaches:

$$\text{MapT}:\langle ID, MElems, KMType, MatchClass, Exp\rangle \qquad \textbf{(equation 1)}$$

- *ID* is the unique identifier of the MapT;
- *MElems* is the pair (a,b) that indicates the mapped elements in the source and destination models;
- *KMType* stands for Knowledge Mapping Type, and is used to identify the morphism as "Conceptual" if mapping concepts or terms; "Structural" if mapping model schemas; or "InstantiableData" if the mapping instantiable properties;
- *MatchClass* stands for Match/Mismatch Classification and is used to classify with reference data, knowledge about the mapping mismatches, i.e., inconsistencies of information that can appear when a mapping between two models is created, derived from the multiple conflicts between the entities;
- *Exp* stands for the mapping expression that translates and further specifies the previous tuple components.

The idea of using a tuple brings many advantages, e.g. being human traceable and readable, adding knowledge concerning mismatch. When used by intelligent systems, the tuple's information enables automatic data transformations and exchange between two organizations working with/on different information models. Therefore, it was decided that the tuple would represent morphism in the framework proposed.

According to the tuple philosophy, all the information about the mappings should be stored in a dedicated knowledge base so that it becomes computer processable, and readjustments can be easier to manage and data exchange re-established automatically

in a sustainable environment. To reach these objectives, Sarraipa et al. [19, 20] proposed the Communication Mediator (CM), and also proposed that all the business partners in the same collaboration network it embedded in their local system.

3.2 Graphical Representation of Mapping Morphisms

In addition to the formalization, also models visualization is important. Frequently information modeling languages are associated to very specific and technically driven graphical representations which damage the abstraction purposed behind modeling.

Graphical browsing of standard models and product data visualization, play important roles in the interoperability achievement, and should be considered in MoMo frameworks. When in the development and implementation stages of an information model, it is frequently necessary to have an easy view and graphical understanding of the full scope of the model. The same happen in the mapping establishment. Thus, non-technical visual representation facilitates the understanding of the reference model, and the abstraction levels that a visual object may represent, brings a suitable and attractive mechanism to understand, navigate and manage the contents of the model, and the model structure itself [21].

For example, nowadays, browsing approaches have been used to assist in the development of some product data standards (e.g. STEP [22]). Efforts towards this kind of visualization were first noticed in XML editors with the introduction of grid layouts. Nevertheless, other more promising technologies exist for these purposes, like hyperbolic tree representation and graph representation. In the first (hyperbolic tree), a tree-like three dimensional hierarchical structure visualization of the information is given, providing the possibility to have represented levels of abstraction with expand/collapse functionalities. Sometimes, despite being technology independent, this type of visualization becomes rapidly complex when models are too large. The second, interactive graph-based representations also do not impose any kind of restrictions on the relationships between the nodes and are considerably more widespread with examples available in many commercial and open source solutions (Microsoft Visio ®, Annas[3], JGraph[4], JUNG[5], etc..)

4 MDA-based Framework for Interoperability Establishment

In order to materialize the vision of being able to put aside the low-level implementation details and have domain experts defining interoperability through the use of language independent information models, a framework based on the four levels of the model-driven architecture, relating meta-models, information models and data is presented in Fig. 2.

[3] https://sites.google.com/site/annasproject/Home
[4] http://www.jgraph.com/jgraph.html
[5] http://jung.sourceforge.net/

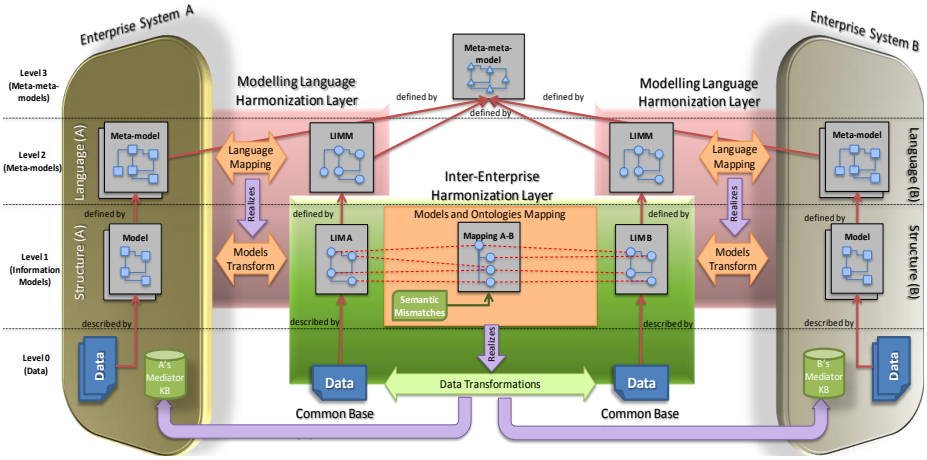

Fig. 2. MDA-based Framework for Interoperability Establishment

The left and right-hand sides of Fig. 2 represent two different organization's information systems with different internal legacy models, where information is presented following the model- language-meta-model. The core of the architecture is focused on the middle part of the figure, enabling two complementary layers, i.e. the modeling language harmonization layer and the inter-enterprise harmonization layer:

- The first (boundaries shared with the enterprises), is focused on the definition of mapping morphisms at the meta-model level, i.e. the modeling language used in each information model. It is therefore the layer realizing the transformation of models from one language to the other, which in our case, is used as an intermediate step for interoperability establishment. Enterprise system models, standards or even reference ontologies are transformed to their abstract interfaces (and vice-versa) using metadata descriptions (the Language Independent Meta-Model - LIMM, presented in next section 4.3) similar to the suggested in ISO/IEC 11179 Metadata Registries (MDR) [23].
- The last (center), works sequentially after the first and is responsible for the model and semantics harmonization, defining mapping morphisms among the different abstract model interfaces (LIMs). The process includes storing this knowledge in a CM knowledge base (as the one of [20]) replicated by the involved organizations, which serving as a standard during the mapping establishment will support the package for sustaining systems interoperability.

The architecture makes use of MDA's horizontal transformations to support the harmonization of modeling languages, models and data levels, within a platform independent context.

4.1 Model Morphisms

Model morphisms are used across the multiple harmonization layers and throughout the MDA levels: Level 2 – language mapping; Level 1 – models and ontologies

mapping, as well as the model transformation morphims; Level 0 – data transformation morphisms.

The MoMo's associated with the mappings are model non-altering"$\theta(A,B)$", which are described by mapping tables for each modelling language linked to the LIMM. These mappings are then implemented using an executable language, realizing the model altering morphisms (transformations "$\tau: A \times \theta \rightarrow B$") on the respective inferior level. Since there are not so many modeling languages available, level 2 mappings are expected to be pre-defined and transformation scripts relatively static as changes in modeling languages specification is not common. They can be updated, but the mechanism for doing so is not envisaged to be as dynamic as the model and ontology mappings from the intra-enterprise harmonization layer (level 1).

4.2 Modeling Language Harmonization Layer

As specified, this architecture layer is responsible for translating information models. Mappings here defined are accomplished by establishing a correspondence, at the meta-model level (level 2 of the MDA), between any specific language constructs and the language independent metadata, enabling bidirectional transformations at any enterprise information model (level 1).

By being able to transform any given input back and forth to the LIM format (LIM meta-model - LIMM), the architecture accomplishes the objective of modeling language independency, helping enterprises to further abstract from technology. To unleash it, executable rules can be applied to transform any N-1 level, according to the N^{th} level of the mapping. This way, one can represent multiple models according to LIMM (level 2) and, if there is a mapping defined between each input modeling language and the latter, multiple models from multiple languages can be represented by equal number language independent models (LIM).

The language mapping procedure is a manual process since meta-models must be analyzed and mapped between them by experts, but the language transformations are always automatic and repeatable. Given that each language map is done only once independently of the number of times it is used or executed, it is an acceptable cost.

4.3 Language Independent Meta-Model (LIMM)

LIMM serves as an abstract interface on top of enterprises' information models. Through its usage, becomes possible to abstract the technology and implementation details associated with the different modeling languages, and thus, enlarge the scope of users involved in a traditional mapping definition activity. Having manager and domains experts involved in this process increases the quality of the mappings that will enable interoperable relationships. In comparison to most modeling languages, it is intended to enable as little loss of expressiveness as possible, but at the same time, be simple and generic to support multiple language mappings.

Also, LIMM resemblances with ISO/IEC 11179 [23] standard are not by fortuity. This abstract interface was based on the standard's foundations and concepts in order to give support to mechanisms for enabling global data interchange, particularly

across application areas. A bridge between major LIMM concepts and ISO/IEC 11179 can be made, e.g. the standard's "Entity", "Property" and "Representation" concepts correspond to LIMM's "Entity_Concept", "Property" and "Representation" constructs, respectively. The language independent meta-model proposed is described as an UML class diagram in Fig. 3.

Many of the information modeling languages, e.g. EXPRESS [24], UML class infrastructure [25], OWL and XSD specification [26, 27] have been analyzed in detail and they were the focus of the attention to create this comprehensive meta-model and, as far the mappings defined for those languages demonstrate, LIMM is able to support them with little loss of expressiveness. In resemblance to what happens in the OWL language, LIMM is capable of representing both models and data levels of MDA (Level 1 and Level 0, respectively), enabling the combined transformation of both levels at the same time, or each independently if required. With this, not only the meta-model is prepared to deal with harmonization of modeling languages, but is also capable of representing instances of models, meaning that it can be used as an intermediate platform for data harmonization (represented by the "LIMM_Instances" package, on the bottom).

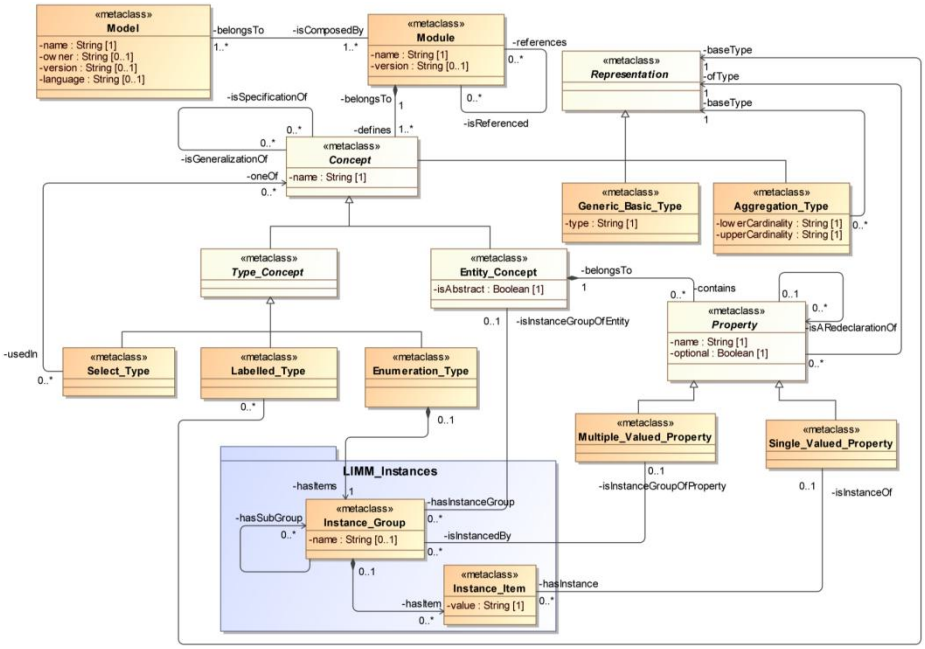

Fig. 3. Language Independent Meta-model (LIMM)

Concerning modeling concepts, the meta-model considers the representation of entities, types, properties, basic types, aggregations, etc. Nevertheless, some languages (e.g. EXPRESS) enable explicit behavioral expressions (instantiation rules) and functions, which are not supported. However, they are considered non-fundamental for the envisaged mapping process which is mainly focused on the information model mapping at the level 1 of the framework.

4.4. Inter-Enterprise Harmonization Layer

Once all modeling languages from the different enterprises involved in the mapping definition are harmonized with the LIMM, and the models made available as LIMs, experts from each company should begin cooperating to define the actual P2P, P2Standard or P2Ontology mapping definition. As specified in the center of Fig. 2, the inter-enterprise harmonization layer is responsible for this activity, following the same MDA horizontal transformation paradigm as before, and enabling automating transformations at the level N-1.

Besides the traditional connectivity, the semantic mismatches, found along the various model elements being mapped, are a very important topic regarding the experts' collaboration. Many of the mapping morphisms will be imperfect due to a number of factors that can go from a simple encoding difference in equivalent properties to a granularity divergence. These can never be solved, but for change management and sustainability this is an important issue and the proposed architecture takes this in consideration, registering the complementarity between the model element correspondence and the semantic mismatch.

The mappings realized at this point do not suffer from the extra complexity of dealing with multi-modeling languages, focusing just on the business related constructs and easing the process of harmonizing the semantic and structure level of models and ontologies. As a result of the entire process, generation of transformation morphisms for data from different enterprise nodes, or even to a reference format, is achieved, thus establishing interoperability.

Each pair of morphims (mappings and transformations) is stored on dedicated Communication Mediators. The objective is that each organization keeps its own CM to track relationships of their inner-elements with its business partner ones, thus maintaining a traceable record of relationships to support monitoring and intelligence activities of the package for sustaining systems interoperability, as well as "on-the-fly" composition of transformations. MoMos defined at the modeling language harmonization layer could also be stored on each CM. However, those transformations are only used to enable the inter-enterprise mapping process, and do not have the same need for dynamicity nor monitoring. The union of the two transformations (for each direction of communication) unleashes the capability of, both automatic and transparently, communicate and collaborate with other organizations, with different modeling languages, models, semantics and ontologies.

The complete automatic data exchange and translation can be accomplished between different model instances at the MDA level 0, thus completing the base for sustainable systems interoperability. Also, since all mappings of level 1 can be stored on a local knowledge base, it enables to gradually add more mappings with other partnering organizations and even to edit or delete past mappings. This provides the required adaptability of the framework to small collaboration networks, and being able to escalate to larger scenarios.

Although the MDA-based framework for interoperability establishment proposes a complete solution to enable the model and language independency in multi-sized business networks, it is more focused in enabling the harmonization of the heterogeneous information models from the multiple organizations involved in the collaboration network. Semantics analysis through terminology mapping is also

possible but, the further refinement of semantic interoperability is not in the scope of this paper.

5. Proof-of-Concept Implementation

Given the context of MDA, QVT is the standard transformation language proposed by OMG. However, considering the languages analyzed, ATL has currently the largest user-base and the most extensive information available such as reference guides, tutorials, programmers' forums, etc. As evidenced by [28], it is a largely used language to implement MDA based tools, having a specific development toolkit plug-in available in open source[6]. By all these reasons it was decided to use ATL to implement model and language transformations in the scope of the MDA-based framework for interoperability establishment.

The proof-of concept (POC) here described, is focused on the implementation details as required by an industrial case-study in the frame of the European Project CRESNDENDO [29], which among other modeling languages is concerned with OWL. This way, Fig. 4 is focused on step required to instantiate the framework previously presented with information models described in that language.

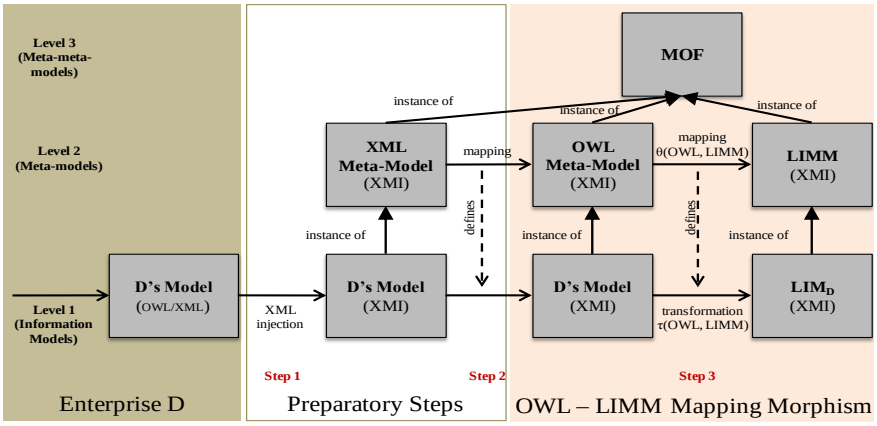

Fig. 4. OWL Instantiation of the Modeling Language Harmonization Layer

5.1 Modeling Language Harmonization Layer

To enable a mapping among the OWL meta-model [30] and the LIMM, one needs firstly to put the OWL data in an XMI serialization following the OWL meta-model specifications. Nevertheless the procedure to do so is not as straightforward as desirable since, in spite of the inputting OWL model is already XML serialized, it cannot be directly processed by the ATL toolkit which needs XMI as an input. The

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

complete process for accomplishing the language mapping test case is illustrated in Fig. 4, where the first step consists in doing an injection of the original model to an XML MOF meta-model specification. Following that, the second preparatory step consists in mapping that XML format to the reference OWL meta-model which will be the starting point for the actual $\theta(OWL, LIMM)$ language mapping (step 3).

According to the architecture specified in section 4, the language transformation is a direct consequence of the mapping. In fact, by using ATL as the MDA language, one is at the same time specifying the mapping and defining the transformation rules (illustrated under step 3).

5.2 Inter-Enterprise Harmonization Layer

LIMM has the unusual capacity of storing both model and data instances within the same physical file, in resemblance to what happens with OWL. It potentiates the actual data transformation at a language independent form as well, thus avoiding the definition of mapping morphisms at this abstract level, which would have to be reengineered back to the original model languages. This integration of model and data maintains a forward flow of activities from company "X" to the abstract interface, and from there to the company "Y". However, as illustrated in Fig. 5, before the definition of the model mapping (step 5), similar steps as the ones conducted for the modeling language harmonization layer need to be followed to append data into the LIM model (step 4). For this TC, since enterprise "A" was already part of the network, that preparatory step 4 is not required.

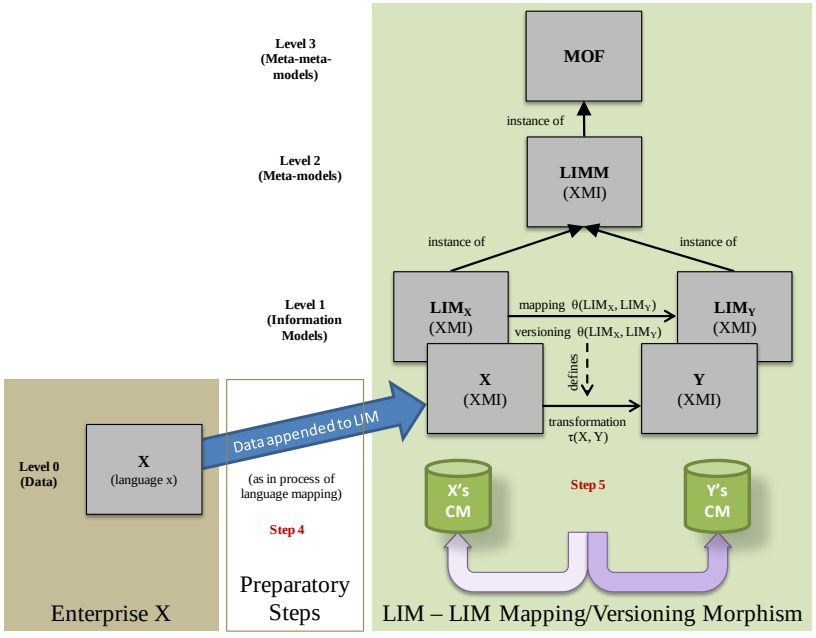

Fig. 5. Instantiation of the Inter-Enterprise Harmonization Layer

It is very important to preserve the user's technology abstraction, envisaged by LIMM, thus the mapping process is supported by a collaborative tool capable of visualizing and interacting with models and concepts in a way that model element' relationships and dependencies are easily understood by domain actors with no knowledge of technical rules. For this purpose, graph-like visualization tools have been analyzed, not being associated with other types of technical diagrams (e.g. UML, etc.). As in the language transformation, data transformation is a direct consequence of the mapping.

5.3 Mapping Tool

JGraph has been elected and modified to read LIM model files and store morphisms at the CM. It is a widely used open source project for graph visualization and manipulation, similar to Microsoft Visio®, with good documentation and several examples. Features include a complete selection of layouts to automatically position the graph, many styles of shapes and edges, validation of connections, as well as an undo and redo manager. Naturally, some adjustments had to be made to enable the interaction (mapping definition) among two different graphs, and to become integrated with LIMM's *Entity_types*, *Type_Concepts*, and *Instance_Groups*. A JAVA binder (JAXB[7]) was included to allow the unmarshalling (interpretation) of LIM files, and JENA[8] - a Java API for OWL providing services for model representation, parsing, database persistence, querying - was used for the integration with the communication mediator.

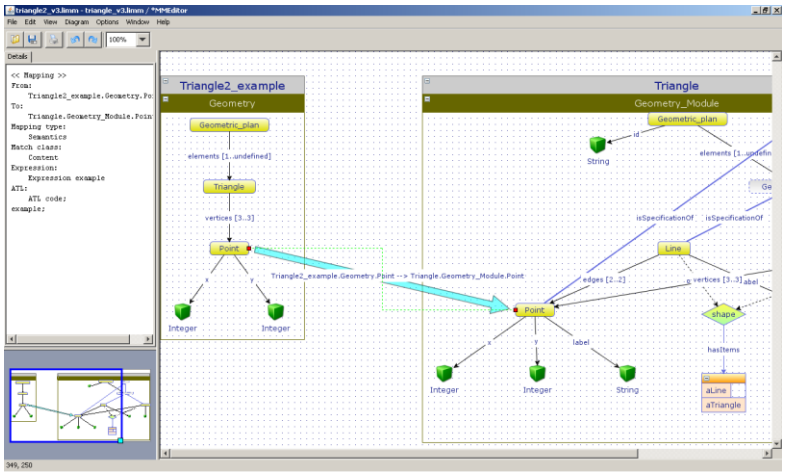

Fig. 6. Mapping Tool Snapshot

A snapshot of the tools in included in Fig. 6 where is possible to see two information models represented using very simple shapes, metadata of the selected

[7] JAXB: http://www.oracle.com/technetwork/articles/javase/index-140168.html
[8] Available at: http://protege.stanford.edu/plugins/owl/jena-integration.html

object on the left, and the mapping linking both model objects. The complete that can be defined between both models is represented not only graphically. It can be edited according to the formalization tuple described in section 3.1 – equation 1, and complemented with the required ATL code.

6. Conclusions and Future work

The proposed conceptual framework envisages that enterprises willing to join a collaboration network do not have to change their legacy software. The choice of MDA/MDI as the enabling technology for the interoperability establishment is motivated by morphisms modularity and repeatability through the existing landscape of tools available to support horizontal and vertical transformations. Depending on the initial situation (i.e. already having a legacy system, or wanting to develop a new one), either of these methods can prove to be the more efficient to establish interoperability, thus allowing a seamless exchange of information among partners.

This branch of applied research could be explored in the future, checking the feasibility of creating smaller, more parameterized software or services developed specifically for managing networked business relationships. Nevertheless, since there are scarce implementations of transformations from context independent models (CIM), where the business requirements are specified, to platform independent models (PIM), where the information structure is detailed, new concepts, methods and tools are demanded to cover this gap.

Acknowledgments. Authors would like to acknowledge the European funded Project UNITE (FP7 248583), namely its secondment programme coordinated by UNINOVA-GRIS, that supported the development of various ideas and concepts presented in this. Also, recognition goes to all the involved in CRESCENDO FP7-234344 and MSEE FP7-284860 that in somehow have contributed to this work.

References

1. Selic, B.: The pragmatics of model-driven development. (2003).
2. Bézivin, J.: Model Driven Engineering: An Emerging Technical Space. GTTSE 2005, Braga, Portugal (2005).
3. Frankel, D.: Model Driven Architecture – Applying MDA to Enterprise Computing. OMG Press (2003).
4. Ambler, S.W.: Agile Model Driven Development Is Good Enough. IEEE Software. 20, 71 - 73 (2003).
5. Czarnecki, K., Helsen, S.: Feature-based survey of model transformation approaches. IBM Systems Journal. 45, 621-645 (2006).
6. Ambler, S.W.: Effective Practices for Modeling and Documentation, http://www.agilemodeling.com/.
7. Thomas, D., Barry, B.M.: Model Driven Development: The Case for Domain Oriented Programming. OOPSLA'03. ACM Press (2003).

8. Greenfield, J., Short, K., Cook, S., Kent, S.: Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. Wiley (2004).
9. MDA Guide Version 1.0.1 (omg/2003-06-01). Object Management Group (2003).
10. OMG: Model Driven Architecture, http://www.omg.org/mda/.
11. Delgado, M.: Harmonisation of STEP and MDA conceptual models using Model Morphisms (MSc Thesis), (2008).
12. Agostinho, C., Correia, F., Jardim-Goncalves, R.: Interoperability of Complex Business Networks by Language Independent Information Models. CE2010. Springer-Verlag, Cracow, Poland (2010).
13. Berre, A.-J., Liu, F., Xu, J., Elvesæter, B.: Model Driven Service Interoperability through Use of Semantic Annotations. I-ESA'09. IEEE, Beijing, China (2009).
14. MSEE: Deliverable D1.1.1: Service concepts, models and method at CIM-PIM-PSM level. MSEE IP (FP7 284860) (2012).
15. INTEROP: Deliverable DTG3.1 (MoMo.2): MoMo Roadmap. INTEROP NoE Project (FP6 IST-1-508011) (2005).
16. Ogren, I.: On Principles for Model-Based Systems Engineering. Systems Engineering Journal. 3, 38-49 (2000).
17. Dauben, J.W.: Georg Cantor: His Mathematics and Philosophy of the Infinite. Harvard University Press (1979).
18. Bernstein, P.A.: Applying Model Management to Classical Meta Data Problems. First Biennial Conference on Innovative Data Systems Research (2003).
19. Sarraipa, J., Jardim-Goncalves, R., Steiger-Garcao, A.: MENTOR: an enabler for interoperable intelligent systems. International Journal of General Systems. 39, 557-573 (2010).
20. Agostinho, C., Sarraipa, J., Goncalves, D., Jardim-Goncalves, R.: Tuple-Based Semantic and Structural Mapping for a Sustainable Interoperability. DOCEIS'11. Springer, Caparica, Portugal (2011).
21. Delgado, M., Agostinho, C., Malo, P., Jardim-Goncalves, R.: A framework for STEP-based harmonization of conceptual models. IEEE IS'06.London UK (2006).
22. ISO TC184/SC4: Standard for the Exchange of Product Data - ISO 10303 (STEP), http://www.tc184-sc4.org, (1994).
23. ISO/IEC: Information Technology - Metadata registries (MDR) - Part 1: Framework (ISO/IEC 11179-1:2004), (2004).
24. ISO TC184/SC4: Industrial automation systems and integration -- Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference manual (ISO 10303-11:2004), (2004).
25. OMG: OMG Unified Modeling LanguageTM (OMG UML), Infrastructure - version 2.4.1, http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/, (2011).
26. W3C: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax, http://www.w3.org/TR/owl2-syntax/, (2009).
27. W3C: XML Schema (XSD), http://www.w3.org/XML/Schema, (2001).
28. Jouault, F., Kurtev, I.: On the interoperability of model-to-model transformation languages. Science of Computer Programming. 68, 114-137 (2007).
29. CRESCENDO IP: Collaborative and Robust Engineering using Simulation Capability Enabling Next Design Optimisation (FP7-234344), 2012.
30. W3C: OWL 2 Web Ontology Language MOF-Based Metamodel, www.w3.org/2007/OWL/wiki/MOF-Based_Metamodel, (2008).