Enterprise Architecture Using the Zachman Framework: A Model Driven Approach

Ali Fatolahi, University of Ottawa, 800 King Edward, P.O. Box 437, Ottawa, Ontario, Canada, K1N 6N5; E-mail: afato092@uottawa.ca
Stéphane S. Somé, University of Ottawa, 800 King Edward, P.O. Box 437, Ottawa, Ontario, Canada, K1N 6N5; E-mail: ssome@uottawa.ca
Timothy C. Lethbridge, University of Ottawa, 800 King Edward, P.O. Box 437, Ottawa, Ontario, Canada, K1N 6N5; E-mail: tcl@site.uottawa.ca

ABSTRACT

Many organizations are interested in building their enterprise architectures using the Zachman framework. They hope to solve the problems of misalignment between business processes and information systems along with gaining a desired level of interoperability and flexibility in their IT environment. However, in most cases the Zachman framework remains as a conceptual framework more than a pragmatic one. This causes a serious doubt as to whether the enterprise could satisfy the motivations of employing the Zachman framework. Model driven architecture (MDA) is addressed in this paper as a framework, which is in a very high synch with the Zachman framework. MDA provides a means of flexible reusable model-driven development environment that is being applied in more and more situations everyday. It is also based on commonplace technologies, which makes it popular amongst software engineers and IT specialists. In this paper, we show that MDA could be the key that opens the world of reality to the Zachman framework. Not only does MDA have everything the Zachman framework needs to be practical, but it also follows the same logical structure and very similar metadata language as the Zachman framework.

Keywords: MDA, Zachman, Architecture, MOF, Conceptual Graphs.

1. INTRODUCTION

The Zachman framework (as firstly introduced Zachman (1987)) is considered as of the major origins of Enterprise Architecture (EA) (Wilton, 2001). According to Schekkerman (2005), 25% of current EA-related activities are being done using the Zachman framework, which is the highest rate amongst all the other frameworks.

The Zachman framework aims at reducing the problems of building information systems without strategic and/or business-related considerations. It categorizes different stakeholders' viewpoints into a fixed set of perspectives through which, everybody could find the exact information he/she is interested in. The framework also captures the knowledge of enterprise via abstracting it into a collection of integrated features.

A key factor of the Zachman framework is that it promises the alignment between business and technology because it provides all stakeholders with the same pattern of information. However, this promise could not be realized without having a mechanism to ensure that different viewpoints have correctly been transformed to each other.

Another important issue is the ability of the Zachman framework to capture the enterprise knowledge in an integrated scheme. In practice, it is very hard to track this ability because modelers use different sets of models with no common background. These problems alongside with other ones (Fatolahi & Shams, 2006) mandate the use of the Zachman framework as a conceptual tool. This means that enterprises could not benefit from all of the advantages of the Zachman framework in practice.

Model driven architecture (MDA) is the OMG's solution to increase model reusability and design-time interoperability. MDA deals with model as an asset rather than a cost. A very important feature of MDA is the facility to transform models among different areas. Not only is it easier to build automatic model mappings in the MDA context, but MDA could also be beneficiary when the model transformation is done manually. MDA provides a collection of popular standards beneath a common philosophy to facilitate the process of quality software design and implementation.

We believe that MDA can be used as a source to supply all the modeling needs of the Zachman framework. This is because MDA:

1. provides a strong approach for model transformation. This means that using MDA, it is possible to build models, which are true transformations of each other or at least very close mappings. This capability makes MDA a good choice to solve the challenge of perspective transforming and tracking within the Zachman framework.
2. provides a broad collection of flexible modeling standards based on a single simple basic notation. Therefore, one can apply MDA-based modeling notations all over the Zachman framework with a high chance to preserve the integrity of the enterprise's knowledge.
3. separates modeling concerns just as the Zachman framework does.
4. levels metamodeling in the same way the Zachman framework does.
5. uses a metamodeling language that could be easily translated to the one used by the Zachman framework.

In this paper we try to address some of the problems regarding the Zachman framework using MDA. Employing the two frameworks with each other, we hope for enterprise architectures that would be well defined and understood by all the stakeholders because of the Zachman framework; ones that would be easily developed and maintained because of the popularity of the MDA framework and its supporting tools.

The rest of this paper is organized as follows. In Sections 2, we introduce the Zachman framework and MDA. In Section 3, we mention related work and discuss the similarities and differences of the current research with the previous ones. Section 4 addresses the adaptability of MDA and the Zachman framework with an emphasis on metamodeling languages and levels. In Section 5, some issues regarding the current research are discussed. At last, we provide a summary along with a plan for future work in Section 6.

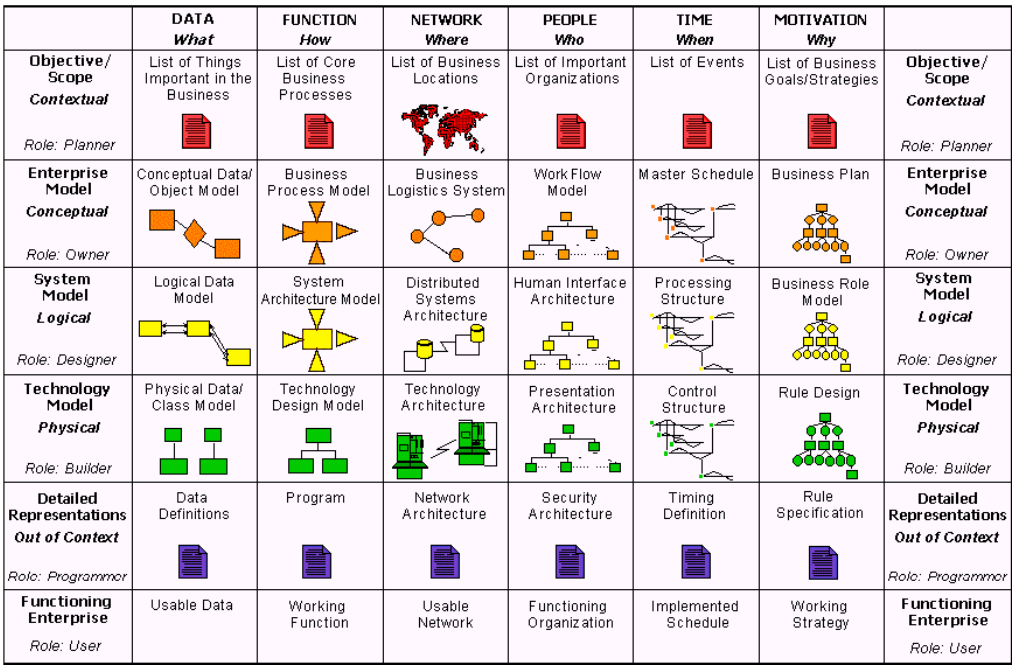
2. BACKGROUND

2.1. The Zachman Framework

Figure 1 is a depiction of the Zachman framework. The Zachman framework consists of six rows and six columns. Rows represent different stakeholders' perspective in building enterprise architecture. Columns are different ways in which, we describe the same product for different purposes. Crossing each row by each column, results in a cell, which contains a unique model.

As it can be seen in Figure 1, the first row is a definition of the context for the enterprise. In the second row, the enterprise is modeled using business modeling techniques. within the third row, the IT environment is conceptually modeled. These design models are mapped onto technology dependent design models in the fourth row. The fifth row contains implementation models. Since systems

Figure 1. The Zachman framework



development is often an outsourced task, which is not performed by the organization itself, this row is supposed to be out of context for the EA. The last row corresponds to the real working enterprise.

Columns of the framework facilitate abstraction of the enterprise's knowledge in a way that is suitable for modeling purposes. Each column is supposed to answer a single question regarding the enterprise. "What are important things for the enterprise?" is answered by the *Data* column. "How does it run?" is answered using the *Function* column. "Where is it located?" is answered in the *Network* column. "Who are acting within the enterprise?" is answered by the *People* column. "When does it perform its businesses?" is answered using the *Time* column and "Why the enterprise does the businesses?" is answered in the *Motivation* column.

2.2. Model Driven Architecture (www.omg.org/mda)

MDA proposes four different layers of modeling. The most top layer is the layer of *Computation-Independent Models* (CIM). CIM represents models, which are valid in spite of the computational options. Business models reside in this layer. Then we have the layer of *Platform-Independent Models* (PIM). PIM acts as a standpoint of systems/software design and architecture. However, it does not contain any information about specific platforms. The third layer, *Platform-Specific Models* (PSM) deals with the technological details of platforms. Here, logical design models are expressed in terms of certain platforms. At the lowest level, there are *Implementation-Specific Models* (ISM[1]). These are real-world objects and components, which act as a running version of the system.

The Meta-Object Facility (MOF) (www.omg.org/mda) is the heart of MDA. MOF provides a means of building new modeling languages and/or transforming different languages each to the other. The MOF is composed of very simple but strong enough elements to describe any other modeling language. Although MOF does not provide any specific notation, it is possible (and convenient) to use basic UML Class modeling notations (with few considerations) to depict MOF models.

MDA admits two levels of MOF-based languages. The first level addresses languages, which are rooted in the MOF itself. In fact, some of these languages such as CWM and UML are even older than MOF, but eventually OMG has refactored them to comply with the MOF. The second level deals with the UML profiles. This level involves different UML extensions. In order to facilitate model exchange amongst different tools and standards, XMI Metadata Interchange (XMI) is also a part of MDA.

3. RELATED WORK

The work closest to ours is reported by (Frankel, et al., 2003). This article shows how different perspectives of the Zachman framework maps to the MDA ones. This is shown in table 1. It also contains a valuable classification of MOF-based models, which can be used to fill in the different cells of the Zachman framework. For example, they propose CWM for the entire *Data* column or *UML Scheduling Profile* for a fraction of the *Time* column. Although we accept this approach, our paper extends the solution with bridging the metadata behind the two frameworks and mapping their hierarchy of metamodeling.

Another interesting work is provided by (B´ezivin & Gerb´, 2001), where the author discusses a precise definition for the MDA framework using CGs. It is worth-mentioning that the paper is published when MOF was not commonplace. The other work is published by (Fatolahi & Shams, 2006), which investigates the capability and popularity of UML models, when applying to the Zachman framework.

In summary, it can be concluded that the related-work has been generally focused on recognizing appropriate models for different parts of the Zachman framework and in the case of (Frankel, et al., 2003) a mapping between the perspectives of Zachman framework and the stages of MDA. Although the work of (B´ezivin & Gerb´, 2001) tries to define MDA using CGs, we do not see an explicit mapping between CGs and MOF in it. Our research is focused on mapping MOF and CGs alongside with discussing the similarity of the approach of the two frameworks towards metamodeling in practice.

Table 1. MDA layers vs. perspectives of the Zachman framework

Perspectives of the Zachman Framework	Layers of MDA
Planner	N/A
Owner	CIM
Designer	PIM
Builder	PSM
Sub-Contractor	ISM
User	N/A

4. ALIGNMENT

4.1. Separation of Concerns

Each row of the Zachman framework could be supplied with models from a certain MDA layer. Our discussion excludes the first row of the Zachman framework. This row represents the planners' viewpoint, which is supposed to be full of textual discriptions of different aspects of enterprise architecture planning, such as constraints, important features, limitations, geographical distribution, etc. We also do not mention the last row that is assigned to the real-working enterprise, which is not related to modeling.

The second row is the owners' perspective. This is the row, which describes different features of the business. This row is the area of different business modeling techniques. Since the models of this row are non-computational, we can use CIM to build the models of this row.

The third row represents the designers' viewpoint. Information systems are designed in this row. However, this design is a platform-independent modeling activity. Models to fill in this row come from PIM.

The standpoint of technology is indicated in the fourth row. In this row, design is simply transformed into technology-dependent modelings. For example, if we chose J2EE as the platform, design *Entities* would become *Entity Beans* here. This is why, we select MDA's PSM as our source for filling this row of the Zachman framework.

Programmers' role is compromised within the fifth row, where the building blocks of the architecture are made up. Because this row deals wholely with the implementation issues, we can feed it using ISM.

4.2. Model Assignment

Although there could be several valid approaches to assign a collection of models to each column of the Zachman framework, the important point is that both (Frankel, et al., 2003) and (Fatolahi & Shams, 2006) show that the family of MDA standards or some subsets of it could cover the Zachman framework. In fact, based on one of the rules of the Zachman framework (Sowa & Zachman, 1992) *there must be a simple, basic and unique model for each column*. Elements of such a model for different columns are provided in table 2. As long as a language can satisfy this rule it could be considered as a valid option. However, there is no guarantee that the chosen language reaches an acceptable level of popularity. Fortunately, this is the main advantage of the MDA, which deals with a family of popular languages.

4.3. MOF vs. CG

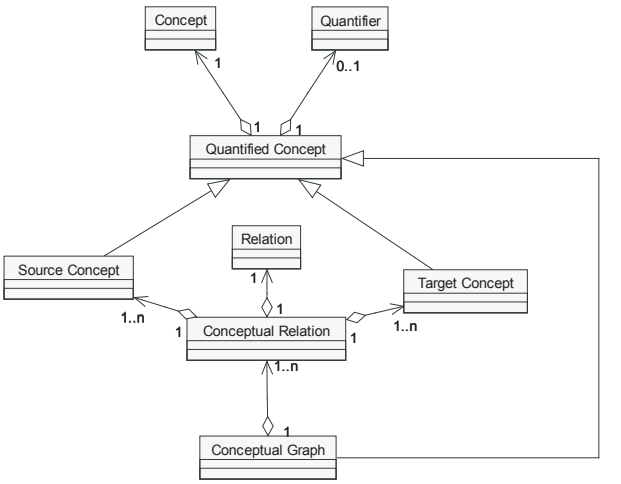
A very important factor to guarantee the quality of final enterprise architecture using the Zachman framework is its integrity. Integrity is gained through applying a set of rules. A question is to know if MDA could facilitate the usage of the Zachman framework. If MDA wanted to facilitate the usage of the Zachman framework, it should support the integrity mechanism of the Zachman framework, which is specified using the metadata language Conceptual Graph (CG) (Sowa, 2000). We address this challenge by providing metamodels showing that the language used to describe each framework could be formally defined using the language of the other framework.

The metadata language of the Zachman framework is CG (Sowa & Zachman, 1992). There is a simple and interesting relationship between CG and MOF. Figure 2 shows how MOF could describe CGs. A *conceptual graph* is composed of some *concept*s and *relation*s. For example, *a cat is on a mat*, is a CG of two concepts, *cat* and *mat*, and a relation, *on*. Usually a concept is being preceded by a quantifier, which is *a* (or more formally *exists*) in this case. Figure 3 is a representation of this CG. Quantifier *a* is not shown because it is the default quantifier. Figure 4 shows how this CG is synthesized using our metamodel. This is a very simple CG but our metamodel is also capable to describe more subtle CGs, including nested CGs. This is done through the generalization association between *Conceptual Graph* and *Quantified Concept* in Figure 2.

Table 2. Essential modeling elements for columns of the Zachman framework

What?	How?	Where?	Who?	When?	Why?
Entity	Function	Node	Agent	Time	Ends
Relation	Argument	Link	Work	Cycle	Means

Figure 2. MOF metamodel for conceptual graphs



We can see that each *Concept* plus an optional *Quantifier* results in a Quantified Concept like "a cat". This could be a target or a source concept. For example, as it can be seen in Figure 4, "a cat" is recognized as a *Source Concept* and "a mat" is realized as a *Target Concept*. A number of source and target concepts then aggregate to a *Conceptual Relation* through a *Relation*, which is "on" in the case of Figure 3 (and Figure 4). Finally, a *Conceptual Graph* is composed of some *Conceptual Relations*.

On the other hand, consider Figure 5 as an essential part of MOF imported from UML metamodel (OMG, 2005). Figure 6, presents a valid CG to express this metamodel. Symbol, *T*, is used as a means of repeating a concept. As it can be seen, multiplicity of MOF associations is shown using quantifier *{*}*, which means a set of concepts. In Figure 6, a "meta" relation represents the generalization from the general type toward the specific type. An "own" relation is used to show the aggregation association among two types.

Figure 3. A cat is on a mat CG

Figure 4. Synthesizing the CG of Figure 3 using the metamodel in Figure 2

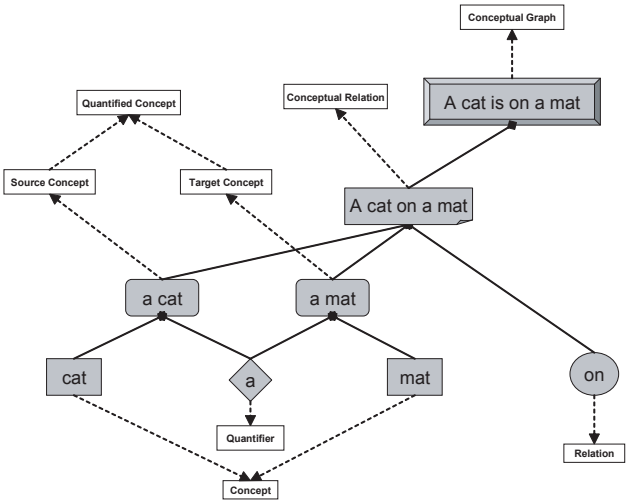


Figure 5. A part of a UML metamodel expressed using MOF

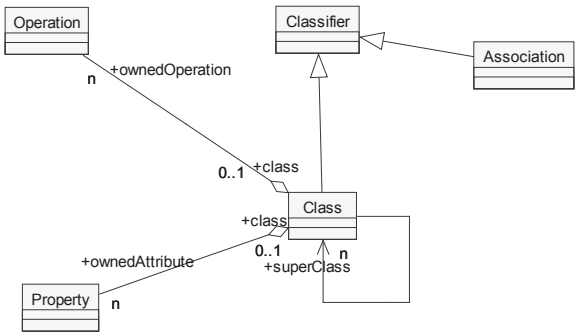


Figure 6. The conceptual graph describing the UML metamodel of Figure 5

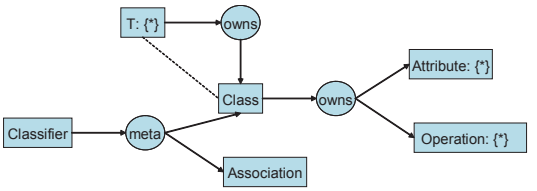
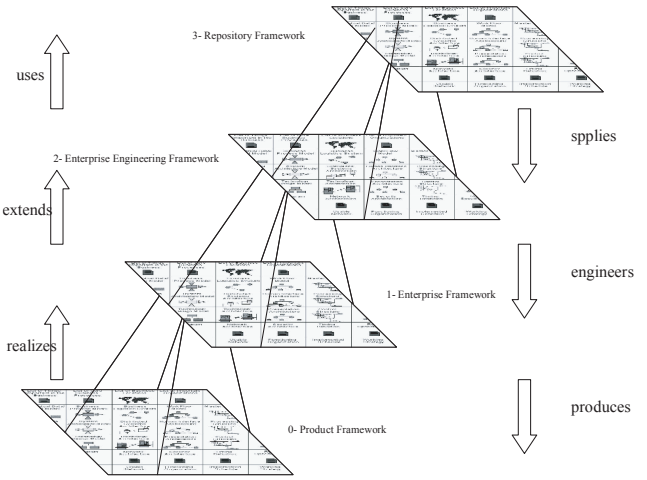


Figure 7. The stack of the Zachman framework



4.4. Metamodeling

An important question would be the ability of MDA to match with the hierarchical structure the Zachman framework provides to maintain reusable and scalable enterprise architectures. We provide general guidelines and ideas to apply MDA at different abstraction levels.

The Zachman framework is not just a pattern to build enterprise architectures; it also provides a mechanism for maintaining the enterprise knowledge. This mechanism guarantees the ability and scalability of the enterprise architecture. This means that the framework makes it possible to change the enterprise at any time using the stack of architectures Figure 7, shows the stack of the Zachman frameworks (Inmon, et al, 1997). MDA has the same mechanism to supply the framework with the models required for each level of this stack. This mechanism is defined within table 3.

According to Figure 7:

1. The framework at level 0 represents the architecture framework for the products of the enterprise. This product could be a software package or an airplane. A product is composed of real working objects, i.e. objects with certain serial numbers. This level could be equivalent to MDA's M0.
2. Architecture for the enterprise itself is modeled within the level 1. The enterprise architecture at level 1 defines the resources and methods through which, the enterprise generates its products. Thus, this level is a supplier for level 0. Since this is the specific enterprise that produces certain products of level 0, architects have to elaborate it using domain-specific models. Therefore, level 1 grabs its modeling essentials from MDA's M1.
3. Level 2 deals with planning, modeling, building and maintaining the enterprise. Enterprise engineers reside at this level. The enterprise engineering framework consists of tools and methods required to define different enterprises. Metamodels are critical for such an activity. Enterprise engineers need metamodels to extend them for the specific aspects of certain enterprises. Using MDA, they can select metamodels from M2. They may extend *UML Activity Model*s to describe the enterprise's workflow or create profiles of *CWM* to define the enterprise's warehousing mechanism (both will then appear at level 1).
4. Finally, there is the Repository Framework at level 3. This is the most abstract framework, which is used to manage enterprise engineering tasks. In fact, enterprise engineers refer to this framework as a general source for all the materials they need in order to develop or maintain an enterprise. Because metamodels are used as a means of enterprise engineering, the repository framework as the supplier of the enterprise engineering framework must also provide a mechanism of handling metamodels. This is why we assign MDA's M3 to this level. MOF is used as the metametamodeling language for all the languages and standards of level 2. It also assures the interoperability of the enterprise models, resulting in everything the Zachman framework promises for: interoperability, flexibility and maintainability.

Table 3. MDA metamodeling mechanism

M3 (MOF)	Metametamodeling layer, including the most abstract materials required to build new languages and interoperability standards.
M2 (UML, CWM, …)	Metamodeling layer, providing the notation and formalism that can be used to model specific domains and systems. This layer is fed by M3.
M1 (User Model)	Projections of M2 in terms of certain user requirements. This includes different extensions of M2 to model the specifications of a certain subject. Examples are UMP profiles.
M0 (Runtime Model)	Runtime objects. Running versions of M1.

Table 4. Mapping the metamodeling mechanism of the Zachman framework and MDA

MDA Metamodeling layer	The Zachman Framework
M3 (MOF)	Repository Framework
M2 (UML, CWM, …)	Enterprise Engineering Framework
M1 (User Model)	Enterprise Framework
M0 (Runtime Model)	Product Framework

Table 4 summarizes the mapping of metamodeling layers of both frameworks.

5. DISCUSSION

Despite of all the similarities, there are some differences between the Zachman framework and MDA. The concept of platform in MDA is different from the similar notion within the Zachman framework. From the viewpoint of the Zachman framework all technological platforms are considered the same and are addressed within the fourth row. For example, Oracle DBMS and .Net framework both belong to a unique category. The case is not the same with MDA, where the architect has to define his/her purpose of the platform explicitly. As a result, a platform-independent model is not necessarily a pure logical model. For example,

a PIM may be independent from .Net but not from the Oracle DBMS; that is the platform means just .Net.

The Zachman framework has a recursive nature. This means that each subset of the framework could be a new framework by itself. For example, as we discussed, the enterprise framework is a product of the Function column of the enterprise engineering framework. MDA does not support this recursiveness explicitly. The mapping of table 4 seems to resolve this problem, but yet the Zachman framework support of the recursive frameworks could be different than the structure of Figure 6.

A possible solution to this conflict is to consider different technologies and standards as members of different columns in the fourth row. For example, we have DB2 in the first column and DCOM in the second column. Since the logic of the Zachman framework is recursive, it is possible to think about each cell independently (Sowa & Zachman, 1992). We could isolate the cell at the fourth row and first column; then restrict the concept of platform to DB2, resulting in *DB2 Specific Model*. We iterate this process within other columns. (e.g. considering DCOM as the platform results in DCOM *Specific Model* for Function column)

6. CONCLUSION AND FUTURE WORK

So far, we have shown that MDA could fit with the Zachman framework very well. Since MDA supports a large collection of standard modeling languages, there is a good chance to assign a subset of MDA to each cell of the Zachman framework. Further, there are many tools and methodologies in use or under development in accordance with MDA.

One of the main benefits of MDA is that it facilitates the process of model transformation. On the other hand, model transformation is one of the major threats against the Zachman framework. MDA makes model transformation easier, more accurate and automated. Here, we find a big set of open problems, which are different MDA mapping functions and transformation paths throughout the Zachman framework.

This research is not just about a conceptual mapping between the Zachman framework and MDA. We believe that these two frameworks could collaborate with each other to build EAs in practice. The purpose of Sections 4.3 and 4.4 was to dismiss the gap between conceptual frameworks and practical applications. According to 4.3, MOF could be translated to CG; so, every model described with MDA is capable to fit within the Zachman framework. The discussion of the Section 4.4 promotes the ability of architects to support all levels of the Zachman framework using MDA.

As our future research, we will mainly focus on different MDA transformation functions to convert and/or track models through adjacent rows of the Zachman framework. Besides, we will be declaring the application of the notion of platform in MDA using the recursive logic of the Zachman framework. Clearly, we will need more and more tools and techniques to support our research. Therefore, this would be an indispensable track of our future research too.

REFERENCES

B´ezivin, J., Gerb´, O. (2001). *Towards a precise definition of the OMG/MDA Framework.* Proceedings of Automated Software Engineering 2001.

Fatolahi, A., Shams, F. (2006). *An Investigation into Applying UML to the Zachman Framework*. Journal of Information Systems Frontiers, Special Issue on Enterprise Architecture, Volume: 8, Issue: 2, 133 – 143.

Frankel, David S., et al. (2003, September). *The Zachman Framework and the OMG's Model Driven Architecture*. Business Process Trends, from Object Management Group database.

Inmon, W., Zachman, J. A., Geiger, J. G.(1997). *Data stores, data warehousing and the Zachman framework, Managing enterprise knowledge*. McGraw-Hill.

OMG. (2005). *Unified Modeling Language Superstructure.* Website: http://www.omg.org/docs/formal/05-07-04.pdf

Schekkerman, J. (2005). *EA Survey Trends 2005 Results.* Website: http://www.enterprise-architecture.info/Images/EA Survey/Enterprise Architecture Survey 2005 IFEAD v10.pdf

Sowa, J. F. (2000). *Knowledge Representation, Logical, Philosphical and Computational Foundations*. Brooks/Cole.

Sowa, J. F., Zachman, J.A. (1992). *Extending and formalizing the framework for information systems architecture.* IBM Systems Journal 31, No. 3, 590-616.

Wilton, David R. (2001). *The Relationship between IT Strategic Planning and Enterprise Architecture Practice*. Journal of BattleField Technology, Vol 4, No. 1, 18-22.

Zachman, John A. (1987). *A framework for information systems architecture.* IBM Systems Journal 26, No. 3, 276-292.

ENDNOTE

[1] This is not an official term from OMG.