View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by AIS Electronic Library (AISeL)

Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2005 Proceedings

Americas Conference on Information Systems (AMCIS)

2005

Mapping the UML to the Zachman Framework

Stevan Mrdalj
Eastern Michigan University, smrdalj@emich.edu

Vladan Jovanovic
Georgia Southern University, vladan@georgiasouthern.edu

Follow this and additional works at: http://aisel.aisnet.org/amcis2005

Recommended Citation
Mrdalj, Stevan and Jovanovic, Vladan, "Mapping the UML to the Zachman Framework" (2005). *AMCIS 2005 Proceedings*. 315.
http://aisel.aisnet.org/amcis2005/315

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2005 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

Mapping the UML to the Zachman Framework

Stevan Mrdalj
Eastern Michigan University
smrdalj@emich.edu

Vladan Jovanovic
Georgia Southern University
vladan@georgiasouthern.edu

ABSTRACT

This article offers an overview of the Zachman Enterprise Architecture Framework (ZEAF) and examines how the Unified Modeling Language (UML) can be used in describing enterprise architecture. The ZEAF is a classification scheme that organizes descriptive representations into a matrix of six distinct stakeholder perspectives and six unique concerns or aspects yielding a normalized approach in which, as a rule, particular cell content cannot be found in more than one cell. This paper presents a comparative review of four approaches for mapping UML onto ZEAF, which despite the above rule, use the same diagram types differently. At first, this appeared to be a problem, but our analysis discovered that it is result of UML's rich and divers notation. This paper also attempts to answer several questions related to these different mappings recognizing an opportunity to extend ZEAF into a multidimensional representation.

Keywords

Zachman Framework, UML, Enterprise Architecture, Documenting Architecture.

INTRODUCTION

Managing enterprise information system architecture is a challenging task as organizations, products, services and technologies continue to change at an increasingly rapid rate. Inspired by architectural engineering, Zachman (1987) and Sowa (1992) created a powerful tool for describing Enterprise Architecture (EA) by combining the views and aspects into a matrix widely known as the Zachman Framework (ZF). The Zachman Enterprise Architecture Framework (www.zifa.com) identifies two different axes of representations which precisely describe the nature and purpose of each cell in the matrix and the deliverables within the organizational context. ZF provided organizations with an effective way of architecting the enterprise information system that confirm to the organization's information needs focusing on ways to ensure a vision-driven development.

Pereira and Sousa (2004) describe the Zachman Framework as one of the most widely known frameworks in the EA context. ZF is a general purpose framework that does not impose a methodology and does not restrict users to a set of pre-defined artifacts. Van Belle (2004) reports on how different authors have applied the ZF to other information system areas which to a greater or lesser extent exhibit similar structural relationships between deliverables, resources, design etc. For example, Bruce (1992) situated information modeling in the ZF, Spewak (1993) used the ZF in enterprise architecture planning and Cook (1996) used the ZF as the basis for building enterprise information architectures. Inmon, Zachman and Geiger (1996) proposed to use the ZF as a guide for the implementation of data warehouses, and De Villiers (2001) even uses the ZF on a meta-analysis level to analyze the framework itself.

The Unified Modeling Language is a general-purpose modeling language originally developed to visually specify the design of object-oriented applications. The complete system model using the UML consists of a set of interrelated diagrams that depict the structure and behavior aspects of the system under design. West, Bittner, and Eddie (2002) well described the strengths of the UML for representing enterprise architecture and its tested means to assess, build, and deploy information systems that "support the organization's core mission" in a cost effective and accurate way. Having the UML as the most commonly used modeling language for object-oriented applications, there is a strong interest on how to use the UML within the EA and specifically for the Zachman Framework as the most widely known architecture framework. Several CASE vendors and practitioners launched efforts to address this challenge. This resulted in different approaches with a noticeable difference in mapping the UML to the Zachman Framework. Different approaches use different diagrams to address the same cell in the framework. Some diagrams address more than one cell in the framework. Another case is when combinations of diagrams address only one cell. The first impression is that this is contrary to Zachman's (1987) normalized approach in which, as a rule, particular cell content cannot be found in more than one cell. Therefore, the generality of structural relationships that is the UML's advantage can be considered a disadvantage leading into confusion about different usage of the same diagrams within the ZF. Additionally, if we do not have a set of unique artifacts for each cell, the expected result

may become distinct and impossible to validate due to the level of dispersion that each user could place. This debate created a need to closely examine the differences in mapping the UML to the Zachman Framework.

The aim of this paper is to present a comparative review of four existing approaches for mapping the UML onto the ZF. The authors are not aware of any similar efforts. This paper also attempts to answer several questions related to the mapping of the UML onto the ZF. The first question is "Why are some diagrams used for more than one cell in the Zachman Framework?" A related question is "How can we bridge different perspectives or aspects using the same diagram types?" The third question is "Can we use more than one diagram to address only one cell in the framework?" The last question is "Is it feasible or, for that matter, necessary that all cells are represented using UML diagrams?" Our analysis discovered that the cause for different mappings is neither the result of inconsistency nor the UML's unfitness to be used for the ZF, instead it is the result of the UML's rich and diverse notation.

THE ZACHMAN FRAMEWORK

Sowa and Zachman (1992) described the Zachman's Framework for Enterprise Architecture (Figure 1) as a matrix that offers taxonomy for relating things in the real world to computer representations: '*The ISA framework serves as a convenient classification scheme or "periodic table" for information entities.*' It provides a means of ensuring that standards for creating the information environment exist and that they are appropriately integrated. The ZF helps govern the architectural process for complex systems with the clarity, dependency, coherence, and traceability needed for an enterprise to manage change and to ensure that the proper alignment is achieved.

Zachman Framework	Data (What)	Function (How)	Network (Where)	People (Who)	Time (When)	Motivation (Why)
Scope (Contextual)	List of things important to the enterprise	List of core processes the business performs	List of locations in which the business operates	List of organizations important to the business	List of events / cycles significant to the business	List of business goals / strategies
Enterprise Model (Conceptual)	e.g., Semantic model Business entity Business relationship	e.g., Business process model Business process Business resources	e.g., Business Logistics system Business locations Business linkage	e.g., Workflow model Organization unit Work product	e.g., Master schedule Business event Business cycle	e.g., Business plan Business objectives Business strategy
System Model (Logical)	e.g., Logical data model Data entity Data relationship	e.g., Application architecture Application function User views	e.g., Distributed system architecture I/S Function Line characteristics	e.g., Human interface architecture Role Deliverable	e.g., Process structure System event Processing cycle	e.g., Business rule model Structured assertion Action assertion
Technology Model (Physical)	e.g., Data Definition Segment/Table/etc. Pointer/Key/etc.	e.g., Program Computer function Data element/set	e.g., Technology architecture Hdw/System structure Line specification	e.g., Presentation Architecture User Screen formats	e.g., Control structure Execute Component cycle	e.g., Rule design Condition Action
Detailed Model (Out of Context)	e.g., Physical storage design Field Address	e.g., Detailed program design Language statement Control block	e.g., Network architecture Address Protocol	e.g., Security architecture Identity Job	e.g., Timing definitions Interrupt Machine cycle	e.g., Rule specification Sub-condition Step
Functioning enterprise	DATA	FUNCTION	NETWORK	ORGANIZATION	SCHEDULE	STRATEGY

Figure 1. The Zachman Enterprise Architecture Framework (Based upon: www.zifa.com/framework.html)

According to Pereira and Sousa (2004) the ZF "proposes a logical structure for classifying and organizing the descriptive representations of an enterprise" by taking into consideration all the participants (stakeholders as users of documentation) involved in the planning, conception, building, using and maintaining of activities of an enterprise information system. It also describes the participant's views by providing a focus on different concerns which apply to each perspective. Each cell stands alone and each is different from the others, although all the descriptions pertain to the same system, and therefore, are related to one another (Goethals, Vandenbulcke and Lemahieu, 2004). Each cell describes artifacts that an organization might document. Cells in the framework are to be considered primitive and therefore, can be modeled or described independently. Zachman (1987) refers to a cell as "normalized" with one fact in one place. The proposed artifacts do not oblige us to follow any type of pre-defined notation. However, a notation should reflect the intention of a cell and enable the direct

correspondence between the cell content and a form of representation (Pereira and Sousa, 2004). The ZF lets us develop a formal framework for maintaining the various documents describing the business requirements that the EA realizes and to support their traceability and linking.

THE UNIFIED MODELING LANGUAGE

The standardization of UML is maintained by the Object Management Group (www.omg.org). The current UML 2.0 version (www.uml.org/#UML2.0) uses 13 diagram types listed in Table 1 (Fowler, 2004). The UML diagram types are not particularly rigid and elements from one diagram type may be occasionally used in another. It is intended to be used pragmatically to reflect the problem and modeling approach taken. The UML descriptions can be used as sketches, detail engineering blueprints or even executable specifications, clearly with a wide range of rigor and details befitting the intent.

	Diagram	Purpose
Structure	Class	Properties and relationships of classes
	Component	Structure and connection of components
	Composite structure	Runtime decomposition of a class
	Deployment	Deployment of artifacts to nodes
	Object	Example configuration of instances
	Package	Compile-time hierarchic structure
Behavior	Activity	Procedural and parallel behavior
	Communication	Interaction between objects with emphases on links
	Interaction overview	Mix of sequence and activity diagrams
	Sequence	Interaction between objects with emphases on sequence
	State	Event changes of an object over its life
	Timing	Interaction between objects with emphases on time
	Use case	User interactions with a system

Table 1. UML 2.0 Diagram Types (Adapted from: Fowler (2004), p.11)

Nowadays, the software development community realizes the UML's potential for much more than visual descriptions of software. West, Bittner and Glenn (2002) state that the UML's applicability and ease of use extends into many domains: business modeling, data modeling, and system modeling. They add that it is "ideal for developing precise and complete visual descriptions of the elements" of enterprise architecture.

A primary strength of the UML is that it is defined by a meta-model. Therefore, the models developed using the UML are potentially consistent in their definitions and can support traceability and linking. West, Bittner, and Eddie (2002) list the following strengths of the UML for representing enterprise architecture:

- The UML is an industry standard notation for developing and documenting systems of all kinds.
- It can help all the way from capturing the business or operational process to defining the EA and specifying systems that support it.
- By providing a single, rigorous notation for documenting all the disparate views of the EA, the UML enables relationships to be more easily represented, communicated, and understood. This facilitates communication and uncovers errors in understanding more easily than other approaches.
- As the UML can be used to define the EA as well as the supporting system, it can also be used to assess compliance of a system with the defined EA.

MAPPING THE UML INTO THE ZACHMAN FRAMEWORK

In this paper, we decided to review the following four empirical approaches of mapping the UML into the ZF developed by major CASE vendors and organizations

- The Rational Unified Process mapped into the ZF (De Villiers, 2003).

- The Popkin Process for Enterprise Architecture which is based upon the ZF and uses the Popkin's tool System Architect (Popkin Software).
- The Zachman Framework applied through the Select Perspective and the UML using the Select Component Architect (Select Business Solutions).
- OMG's Model Driven Architecture mapping to the ZF (Frankel, Harmon, Mukerji, Odell, Owen, Rivitt, Rosen, and Soley, 2003).

There are several other approaches, like Moriarty's (2001), but we deliberately restricted our attention here to the approaches that are either supported by the aforementioned organizations or that have good documentation and are generally available.

Our comparative analysis of the different mappings is organized by the ZF perspectives: Scope, Enterprise Model, System Model, Technology Model and Detailed Representation. Definitions of these perspectives are provided later. Since the Operational perspective is a view of the functional system in its operational environment, there is no need to represent artifacts in this row with any UML diagrams. That leaves us with 30 cells for which we will provide a review of the used artifacts. In this paper, the UML artifact means any kind of UML representation, model or diagram, which supports the cell's intention.

Planner's View

This view describes the business purpose and strategy, which defines the extent of the other views (Table 2). De Villiers (2001) stated that this view "serves as the context within which the other views will be derived and managed."

		Data (What)	Function (How)	Network (Where)	People (Who)	Time (When)	Motivation (Why)
Scope	**Zachman**	List of things important to the enterprise	List of core processes the enterprise performs	List of locations where the enterprise operates	List of organizations important to the enterprise	List of business events / cycles	List of business goals / strategies
	Rational RUP	Class (Business entities)				Activity (Business workflow)	
	Popkin						
	Select	Package			Activity (Business actors)	Activity (Events)	
	OMG MDA	Package, Class, Use Case	Activity, Use Case				

Table 2. Scope Artifacts

Diagrams are not necessarily the best ways to communicate the Scope perspective, which identifies the enterprise at a high level. Moriarty (2001) suggested that diagrams "may be too structured to meet the needs of the intended audience." The Popkin Process (www.popkin.com) also argues that UML diagrams are not appropriate for this view. When UML diagrams are used to describe the scope, expect to see some variation of the UML diagrams like in OMG's usage of the Use Case diagrams to represent business entities.

Owner's View

The owner is interested in the business deliverable and how it will be used. This provides a description of the organization within which the information system must function. The artifacts from this view provide an understanding about the areas of the enterprise that should be automated (Table 3). Since the emphasis of the Owner's view is on what the business does and what it needs to know about, the question is if UML diagrams are useful for developing business models. Moriarty (2001) suggested that they are, when the elements in the model are named "with terms that are meaningful to the business" in plain English. For example, this can be done using the UML stereotype <<business>> for classes, actors, workers or use case. Another example is to name the swim lanes in the activity and interaction diagrams using business roles such as vendor, account specialist or marketing analyst.

		Data (What)	Function (How)	Network (Where)	People (Who)	Time (When)	Motivation (Why)
Enterprise Model (Conceptual)	**Zachman**	Conceptual data model	Business process model	Logistics network (nodes and links)	Organization chart, workflow model	Business master schedule	Business plan
	Rational RUP	Class (Business classes)	Use Case (Business Use cases)		Use Case (Business Actors)	Use Case (Events)	
	Popkin		Activity				
	Select	Class (Business classes)	Activity (Process flow)		Activity (Business Actors)	Activity (Events)	
	OMG MDA	Class	Activity, State, Sequence, Communication				

Table 3. Enterprise Model Artifacts

Designer's View

This view outlines how the system will satisfy the organization's information needs. This view should not contain solution-specific details or production-specific constraints. The specifications in this view should ensure that the system will fulfill the owner's expectations (Table 4).

		Data (What)	Function (How)	Network (Where)	People (Who)	Time (When)	Motivation (Why)
System Model (Logical)	**Zachman**	Logical data model	Application architecture	Distributed system architecture	Human interface architecture	Dependency diagram, entity life history	Business rule model
	Rational RUP	Class (Persistent classes)	Use Case (Realization)	Deployment	Class (Boundary classes)	Sequence, Communication	Class (Multiplicities)
	Popkin	Class	Use Case	Component	Use Case		
	Select	Class	Use Case	Deployment, Component	Sequence, Class	State	
	OMG MDA	Class, Package, Component	Activity, State, Sequence, Communication	Deployment			

Table 4. System Model Artifacts

Diagrams from the logical perspective are more technical with the elements usually stereotyped based on the role they play in the application architecture. For example, the persistent stereotype represents those classes that must persist over time, the control stereotype represents those classes that synchronize the interactions between objects, and the boundary stereotype is used to represent the user interface objects (Moriarty, 2001).

Builder's View

The builder manages the process of assembling and fabricating the components in the production of the product. This view addresses production constraints and provides a specification of how the information system will be implemented. It emphasizes the specific solutions and information technologies (Table 5). Diagrams used within the technology perspective are quite similar to those used in the logic perspective with more emphasis on the behavior and implementation details.

Subcontractor's View

The subcontractor fabricates out-of-context components which meet the builder's specifications. This view puts more stress on the parts of the system versus the whole system architecture. It is concerned with implementation-specific details of the individual system elements (Table 6).

		Data (What)	Function (How)	Network (Where)	People (Who)	Time (When)	Motivation (Why)
Technology Model (Physical)	**Zachman**	Physical data architecture	System design	System architecture	Presentation Architecture, User interface	Control structure	Business rule design
	Rational RUP	Class	Component				
	Popkin	Class	Use Case, Activity	Deployment		Sequence, Communication, State	
	Select		Use Case, Class, Sequence, Component	Component, Deployment	Sequence, Class	Sequence, Communication	Package, Class, Sequence, Communication
	OMG MDA	Class, Package, Component	Activity, State, Sequence, Communication	Deployment			

Table 5. Technology Model Artifacts

The use of diagrams becomes less and less important as we progress from the technology to the out-of-context perspective and begin to approach the code. For instance, the logic of some method in an object class should correspond to the rules for state transitions existing in a state chart for that class. Another example is the database schema which should contain all persistent classes. Similarly, technologies such as COM or CORBA are used to implement the interaction of the objects across the network. Therefore, just a few diagrams are allocated to this perspective of the Zachman Framework.

		Data (What)	Function (How)	Network (Where)	People (Who)	Time (When)	Motivation (Why)
Detailed Model (Out of Context)	**Zachman**	Physical storage design	Detailed program design	Network architecture	Security architecture	Timing definitions	Rule specification
	Rational RUP					State	
	Popkin						
	Select			Component, Deployment		Sequence	Sequence
	OMG MDA						

Table 6. Detail Representation Artifacts

THE CONVERGENCE

In original Zachman works the framework defines that framework cells contain single primitives (artifacts.) Such a normalized approach governs that particular cell content cannot be found in more than one cell. This leads into an assumption that UML diagrams should be consistently mapped onto the Zachman Framework. As can be seen above, there is a noticeable difference in doing so. Let us first provide a comparative analysis of the diagrams used to represent the different aspects of the system.

Things - The content of the systems is consistently described using Class diagrams. Package diagrams are used for grouping or hierarchical structuring. Component diagrams are also used at the logical and physical level to represent the intended structural representation.

Process - The functional aspect of the system is also consistently described using Use Case and Activity diagrams at the scope and conceptual views. At the logical and physical views, the Sequence and Communication diagrams are used to describe the interaction between objects. Class diagrams can also be used here to emphasize an object's behavioral details like visibility etc. Similarly, component diagrams can be used to indicate the required interface. Occasionally, State diagrams can be used to indicate activities that take place during the object's state changes.

Locations - There is no specific UML diagram that can be used for Zachman's network aspect. Still, there is a need to show that the business object classes are utilized at various locations where business is conducted. Deployment diagrams are used to map those location dependent classes into the network nodes. To some extent, Component diagrams can be used to show where the component would reside.

People – The UML does not provide a specific diagram to indicate people or organizations important to the business. Instead, the actors used in Activity, Use Case, Sequence and Communication diagrams represent the roles people or organizations assume in the business. Class diagrams can also be used for this purpose if, for example, they indicate boundary classes.

Time – Although the UML 2.0 provides the Timing diagram it is not used to represent sequencing and timing of the processes and flows in the Zachman Framework. This is partially due to the fact that this diagram is added to the UML after the methodologies considered here are created. Instead, the Activity and Use case diagrams are used to show events at the scope and conceptual views. More detailed sequencing and timing is indicated using the Sequence, Communication and State diagrams.

Motivation - No specific UML diagrams exist to represent the enterprise's business rules or to specify how the information system assemblies support those business rules. Instead, the business rules are expressed as various constraints or adornments in all UML diagrams and specifically in the Class, Sequence and Communication diagrams.

Base on the above analysis, our first question is "Why are some diagrams used for more than one cell in the Zachman Framework?" This is especially true across the aspects. For example, Class diagrams are used in almost all cells of the "Data" aspect as illustrated in Figure 2. The second observation is that the same diagram types are not only used across the aspects, but they are used across perspectives for different aspects as shown in Figure 3.

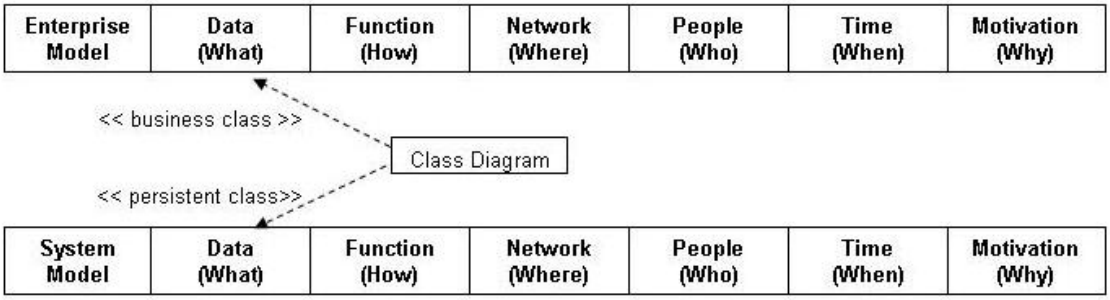

Figure 2. Bridging the Zachman Framework perspectives

The answer to this question is that it is because UML diagrams are composite models using several primitives in the same diagram that are classified in different cells in the Zachman framework. Therefore, it is acceptable that some diagrams address more than one cell in the Zachman Framework — sometimes disjoint cells.

The separation of concerns as a principle provides for a clear framework for conceptualization and selection of technologies, but it is not supposed to preclude people from thinking in terms of 'unexplored' relationships leading to more integrated systems. A related question is "How can we bridge different perspectives or aspects using the same diagram types?" Figure 2 illustrates how bridging of the framework's rows can be achieved using stereotypes, adornments and icons to provide a different perspective. Moriarty (2001) explains this ability to use a diagram across rows by employing different UML features within each perspective.

Similarly, Figure 3 illustrates how the Activity diagram describes workflows in terms of activities, roles (actors), events and objects. This also implies that relationships between the *What, Who*, *When,* and *How* columns cannot be represented within the Zachman Framework (De Villiers, 2001).

Another example is the Class diagram which expresses the concept of an object in terms of the interrogatives it addresses and the perspective it serves. If, for example, you consider an object to be the encapsulation of data, processes, and rules that it plays, then a deliverable comprising such objects fits into row 3, columns 1,2, and 6 (von Halle, 1997). Similar attempts to

further extend the ZF to the Object-Oriented Framework are also addressed by Bruce (1992). It is simple, but true, that humans may have needs to relate different concepts from different aspects, especially in the case of cross-cutting concerns, and that stifling innovation in the sense of preventing possible combinations from being viewed can be counter-productive.

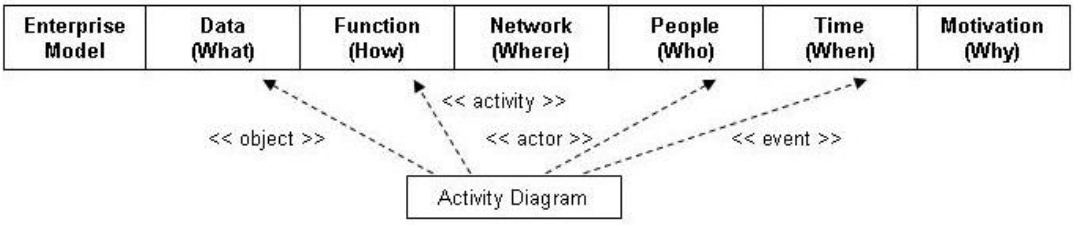

Figure 3. Bridging the Zachman Framework aspects

Our next observation, that the combinations of diagrams address only one cell, leads us into the third question "Can we use more than one diagram to address only one cell in the framework?" The answer is yes, due to the fact that different UML diagrams contain the same primitives. For example, events can be expressed in the Activity and State diagrams. Therefore, the same event can be documented with different types of diagrams or some events can be documented with one type of diagram and other events can be documented with another type of diagram.

It is also easy to see that there are cells in the Zachman Framework which have no UML coverage. This creates our last question "Is it feasible or for that matter necessary that all cells are represented using UML diagrams?" We may find the answer to this question in the following common myth about the use of the Zachman Framework (von Halle, 1997):

Myth 1. You must deliver components for every cell.	**Truth 1.** You should select only those cells whose deliverables will assist in achieving specific business goals and in integrating the enterprise. Every cell costs money to deliver and manage.

As we have indicated before, no special diagram is necessary to communicate the enterprise's mission, vision statement, goals, strategies, tactics, and plans. On the other hand, the UML still has the greatest potential for displaying correlated elements of interest to various stakeholders.

Contrary to the initial assumption, the answers to the questions reveal that the cause for different mappings is neither the result of inconsistency among different mapping approaches nor inability of the UML to be effectively used for the ZF, instead it is a result of UML diagrams being composite models and the UML's rich and diverse notation.

CONCLUSION

The Zachman's Framework for Enterprise Architecture is a well known tool used to define what products must be delivered to whom (stakeholder view) and what concerns and subject matter they are addressing (aspects). In this paper we presented a comparative analysis of four approaches that use the UML to describe systems in the scope of the Zachman Framework. This analysis revealed that there are noticeable differences in how the UML is used to describe framework cells.

The main contribution of this paper is that it debunked the assumption that there should be a unique mapping of the UML into the ZF. This paper provided explanations why it is allowable to have some diagrams used for more than one cell in the Zachman Framework. It illustrated how we can bridge different perspectives or aspects using the same UML diagram types or how to use more than one diagram to address only one cell in the framework. As this paper also demonstrated, there are open dimensions in using the UML for the Zachman Framework. Namely, no viewpoint and concern uniquely stipulates a single possible way to use only one representation even if we focus on the most comprehensive modeling language such as the UML.

It is important to understand here that a particular choice of UML mapping to the ZF still depends on specific organizational needs and requirements. The organizations are still responsible for developing their own know-how for the EA, since consistent and comprehensive total approaches might just be too elusive. In this respect, this paper provides guidance in selecting elements from the UML to develop the organizational instantiation of a Zachman Framework. With the provided analysis it is possible to understand the different possibilities and to recognize the vast potential for the diverse usage of the UML diagrams for documenting framework cells.

The findings from this study helped us to recognize the need for an extension of the ZF with dimensions which would include various representation options. This creates an opportunity to question the ZF rules (Sowa and Zachman, 1992) of preserving the orthogonal and discrete nature of the ZF classification. It is clear that the multidimensional aspect of the ZF has a

potential to contribute significantly to the EA and, above all, to the understanding of the architectural challenges and the required robust design process beyond the representational views and artifacts.

REFERENCES

1. Bruce, T. (1992) Designing Quality Databases with IDEF1X Information Models, Dorset House Publishing, New York.
2. Cook, M. A. (1996) Building Enterprise Information Architectures: Reengineering Information Systems, Prentice-Hall, Upper Saddle River.
3. De Villiers, D.J. (2001) Using the Zachman Framework to Assess the Rational Unified Process. *The Rational Edge*, March 2001. Available online www-128.ibm.com/developerworks/rational/library/372.html (Accessed 01-26-2005).
4. Frankel D.S., Harmon, P., Mukerji, J., Odell, J., Owen, M., Rivitt, P., Rosen, M. and Soley, R.M. (2003) The Zachman Framework and the OMG's Model Driven Architecture, *Business Process Trends*, September 02, Available online www.bptrends.com.
5. Fowler, M. (2004) UML Distilled 3rd ed. Addison Wesley, Boston.
6. Goethals, F., Vandenbulcke, J. and Lemahieu, W. (2004) Developing the Extended Enterprise with the FADEE, *Proceedings of the ACM Symposium on Applied Computing*, March 14-17, 2004, Nicosia, Cyprus, 1372-1379.
7. Inmon, W.H.; Zachman, J. and Geiger, J. (1997) Data Stores, Data Warehousing and the Zachman Framework: Managing Enterprise Knowledge, McGraw-Hill, New York.
8. Moriarty, T. (2001) To Unify Architecture with Methodology: The Rational Unified Process meets the Zachman Information Systems architecture, *Intelligent Enterprise*, April 2001, Available online www.intelligententerprise.com /010416/print/metaprise.jhtml.
9. Pereira C.M., and Sousa, P. (2004) A method to define an Enterprise Architecture using the Zachman Framework, *Proceedings of the 2004 ACM symposium on Applied Computing*, Nicosia, Cyprus , 1366 – 1371
10. Select Business Solutions, Zachman framework to Select products mapping, www.selectbs.com/products/solutions /zachman_framework.htm.
11. Sowa, J.F. and Zachman, J.A. (1992) Extending and Formalizing the Framework for Information Systems Architecture, *IBM Systems Business Journal*, 31, 3, 590-616.
12. Spewek, S.H. (1993) Enterprise Architecture Planning, QED Publishing Group, Boston.
13. West, D., Bittner, K. and Eddie Glenn E. (2002) Ingredients for Building Effective Enterprise Architectures, *Rational Edge*, November 2002, Available online www-106.ibm.com/developerworks/rational/library/content/ RationalEdge/nov02/Enterprise Architectures_TheRationalEdge_Nov2002.pdf
14. Van Belle, J-P. (2004) Leveraging IS theory by exploiting the isomorphism between different research areas, *Proceedings of the 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, Stellenbosch, Western Cape, South Africa, 107 - 114
15. Von Halle, B. (1997) What IS can and should learn from traditional architecture: Architecting in a Virtual World, *Database Programming & Design On-Line*, Available online www.dbpd.com/vault/9611arch.htm
16. Zachman, J. (1987) A Framework for Information Systems Architecture. *IBM Systems Journal*, 26, 3, 276-292.