Fall 2006

Scaling the Zachman Framework a Software Development Methodology for Non-Enterprise Applications

Carla L. Thompson
Regis University

Regis University
School for Professional Studies Graduate Programs
Final Project/Thesis

Disclaimer

Running head: SCALING THE ZACHMAN FRAMEWORK

Scaling the Zachman Framework

A Software Development Methodology for Non-Enterprise Applications

Carla L. Thompson

Regis University

School for Professional Studies

Master of Science in Computer Information Technology

Acknowledgements

First, I would like to thank my dear friends and family for their consistent support during my graduate journey. Your words of encouragement during the weeks of late night development efforts helped me hold the vision of graduation and stay true to my course.

Second, I would like to thank my academic advisor, my thesis advisor, and the facilitators of the Professional Project Courses. Thank you for reflecting excellence and demonstrating your technical expertise. Your professional presence in the Information Technology community has been both challenging and rewarding to strive for.

Finally, I would like to thank Regis University for creating the School for Professional Studies curriculum. This program allowed me to obtain my Masters degree while working full time. I give thanks for the opportunity to earn a Masters Degree while learning the skills needed to stay competitive in the field of Information Technology.

List of Tables and Figures

Table of Contents

Abstract

The software development methodology brought forth in John Zachman's System Architecture Framework can be used to design a wide range of information systems. While the Zachman Framework is very robust and typically used for developing large scale Enterprise Applications, this project will demonstrate that the framework can easily be scaled to fit a small scale non-Enterprise Application. The Zachman Framework is a six by six matrix that breaks down system requirements into cells that document the system. Each row of the Zachman Framework will be examined to determine if the documentation would be needed. A discussion about the appropriateness of using the Zachman Framework as the software development life cycle methodology for a small non-Enterprise application will conclude this investigation.

Chapter 1: Project Introduction

1.1 Introduction

By using the software development methodology brought forth in John Zachman's System Architecture Framework, this project will demonstrate how this methodology can be used to design a small scale non-Enterprise Web Application. During the effort of developing an application the system analyst carries the responsibility of finding the best system software design solution. If it is feasible to build a software solution instead of purchasing one off the shelf, a strong consideration should be given to using the Zachman Framework as the software development methodology. There are many software development life cycle methodologies available, however they all lack the ability to provide a framework in which all elements of the system design documentation can be integrated together. The selected methodology should be comprehensive while allowing the project to stay within the development budget. While the Zachman Framework is very robust and typically used for developing large scale Enterprise Applications, this project will demonstrate that the framework can easily be scaled to fit a project of any size.

1.2 Current System

The current system that will be used during this project is an Internet based donation system. This web based application that is used during fund raising efforts to collect credit card donations on the Internet. The system is considered small in size and not part of a larger Enterprise application. The target market for this system would be the English speaking population of the United States. The analysis and development resources for this project are small. Only one resource has been allocated to carry out the full system life cycle for this application. Since the system was developed without using a software development life cycle

methodology, there is no documentation detailing the existing system. Enhanced capabilities may be needed in future implementations of this system. The current system is in need of restructuring. The decision must be made to either purchase a Commercial Off the Shelf product or to take on the challenge of developing the software. If the software is written in house, an extensible software development methodology should be used. The donation system was used to evaluate the ability to apply the proposed software development life cycle methodology.

1.3 Statement of Project Goals

- Determine if the Zachman Framework scalable.
- Determine what components of the Zachman Framework are scalable?
- Determine how the Zachman Framework can be used to provide a development structure for a non-Enterprise Application.

The goal of this project was to prove that the Zachman Framework is scalable. The Zachman Framework is a six by six matrix that breaks down the system requirements into cells that document the system. Each row of the Zachman Framework was examined to determine if the documentation would be needed. A discussion about the appropriateness of using the Zachman Framework as the software development life cycle methodology for a small non-Enterprise application concludes this investigation.

1.4 Statement of Project Scope

Typically Zachman's framework is used to provide structure when developing large scale Enterprise based applications. While applications of all sizes should have development goals that result in a system engineered for scalability, adaptability and extensibility, developers tend not to use a framework such as Zachman's when developing smaller, non-Enterprise web applications. The scope of this project will be to demonstrate how Zachman's work can be an

effective methodology used to drive the analysis process of medium to small applications that function as non-Enterprise based applications. This project did not include a build of the example system. The focus was placed on the ability to scale the Zachman Framework and suggestions are made regarding which documents, charts, models, etc. would be used if the Analysis Phase of the system were carried out.

Chapter 2: Research

2.1 Overview

As the author began this research effort, there were several critical questions that needed to be answered before she could choose a software system solution. The first question was: Should the software be purchased or built/developed in house? If a software build was necessary, what software development life cycle methodology would be used? The author reviewed the concept of "Buy vs. Build". Since a software build was determined to be necessary, a software development methodology was also selected. Several software development methodologies were weighed against each other as possible solutions. The best method was selected.

The following software development life cycle methodologies were reviewed for use with this project: Zachman's Framework, Waterfall and the Iterative and Incremental Model. Each of these methodologies has a different focus. Zachman's Framework has been successful in managing many enterprise scale technology initiatives. Waterfall methodology is used for projects of all sizes. It is considered a top down approach to software development. Iterative and incremental methodology is also used for projects of all sizes. This method uses the Waterfall methodology and applies it to smaller segments of the project. Each segment is revisited and treated as a "mini project". One methodology was selected to be used with this project. After choosing a methodology, the author discusses the contributions this project made to the field of software development.

2.2 Buy vs. Build

2.2.1 Buy Software

As we move forward in the field in Information Technology, there comes a time where we must answer the question: Should we build or buy the software to fill the business need?

There are many components that play a key role in making this decision. As the reusable concept of software development continues to grow, so has the number of Commercial-Off-the-Shelf (COTS) products that are available today. The goal is to maximize efficiency by delivering the best software solution available while staying within resource constraints of the project. In doing so, it is important to always consider all options before finalizing a decision.

Purchasing an existing system that is designed to meet the core needs of a specific business function can save on time and money in terms of development. Assuming this is not a stand alone system, there are still costs associated with fitting the purchased system into the existing system, seamlessly connecting the two so that they work in conjunction with each other.

> In many COTS products, the vendor has defined an architecture that may or may not match the architecture of the purchaser's domain. For organizations that have mature business processes and legacy systems in place it is unlikely the vendor's architecture will match (Alleman, 2002, pg. 1).

When giving the choice between purchasing a Commercial-Off-the-Shelf product or taking on the task to develop the system in-house, the author examined several elements before making the best decision. Initial concerns would include the size of the application that needs to be built, the pending deadlines that might exist to have the project delivered, the availability of a COTS product, and the type of expertise that exists in house.

The author has found using a COTS product to be beneficial in one sense; yet, the purchased system rarely addresses all of the specific needs of the business user. Additional reporting features and functionality identified as major requirements must be present to meet the end users requests. If the COTS product were used without additional modifications, this would be an ideal situation, but is not always realistic. Adding additional customization (i.e. Reporting and proprietary functionality) to the purchase package consumes precious resources. Time is spent first learning the features of the purchased system, then an estimate is made stating the

amount of time and technical resources that will be needed to make the additional changes that are required. All of these changes along with the purchase of the software itself must come in under budget to justify purchasing over developing in house.

2.2.2 Build Software

Depending on the size of the project and the pending deadlines, it may actually be more feasible to build the application in house. The decision to build allows the developers to gain expertise with the functionality from the ground up. The features and functions of the application are built to specification and do not include unnecessary features that could consume system resources or add to the complexity of understanding the application for the end user. When software is developed in house, the developers have the liberty of being able to customize the source coding as needed when their original specifications change without being locked into a purchased solution that now no longer is a fair candidate for solving their business problem.

For this project, the size of the application is medium to small. The user interface will definitely need to be customizable and the reporting needs are very specific to this application. The current application requires that a sponsor making a donation will be provided with an easy and secure way to make that donation using web based technology. There are shopping cart applications that could have been used to generically provide a way to add “items” to the cart. One type of item could have been a donation of a specific dollar amount. While this is a reasonable alternative, it is not what was described as the requirement. If this option were desirable, it would be available through different Internet Service Providers (ISPs). The look and feel of the shopping cart experience would be decided upon by the Internet Service Provider. To maintain full ability to customize the look and feel of the whole web donation experience, it was ideal to build this application instead of purchase an existing product off the shelf.

Time constraints and technical resources are other considerations that would determine whether the application should take advantage of COTS or be a candidate for software development. After further examination, the decision to purchase rather than building was swayed since the look and feel would be the main compromise to have immediate access to the existing shopping cart functionality offered by ISPs. In this case, the need for full customizability far outweighed any pending deadlines, so building the application was desired.

2.2.3 Decision

With the consideration to the size of the project, the custom reports and the need to control the look and feel of the application, it was most efficient and desirable to move forward with the decision to build of the example software rather than opting for commercially supplied alternatives.

2.3 Methodology

2.3.1 The Zachman Framework

In 1987, John A. Zachman published the Zachman Framework for Enterprise Architecture. "The Framework organizes and categorizes architectural representations in a way that provides a context for understanding the relationship between and among separate sets of architecture (Hokel, 2006, pg. 1)." In the past, design artifacts such as business models, organizational charts and the like were created during the system development life cycle. While these charts and models had their individual value, it is more valuable to show these models in an integrated fashion. The Zachman Framework provides a methodology to understand the relationship between all of the models and charts used to design a business system.

The ideas that went into the development of Zachman's Framework were a combination of classical architecture and aircraft manufacturing analogs. Both classical architecture and aircraft

manufacturing have proven track records when it comes to capturing complex constructs. Zachman applied these engineering principals to the framework he developed for information systems architecture.

When developing or architecting a business system solution, different representations of that system are critical to capturing the essence and detail needed for accurate and timely implementation. "The primary strength of the Zachman Framework is that it explicitly shows that many views need to be addressed by architecture (Inmon, Zachman, Geiger, 1997, pg. 129)." Zachman captures these different views in a six by six matrix. The six different perspectives are normally defined as: (Planner)/Scope, (Owner)/Business Model, (Designer)/System Model, (Builder)/Technology Model, (Sub-contractor)/Components and Functioning Enterprise. These represent the six rows that make up Zachman's matrix. The idea here is to use these perspectives to evaluate the system being developed.

Each perspective addresses a fundamental question: What? How? Where? Who? When? and Why? So, the previous six questions will be asked while considering the system's Scope. The same six questions will be asked while considering the system's business model and so on. The focus or fundamental questions make up the vertical columns of this matrix. To quote John A. Zachman, "By abstracting out one of these six variables and holding the other five constant, but not losing sight of them, attention can be intently focused on only one variable (Hokel & Thomas, 2006, pg. 2)." So, by answering all of these questions while considering the six different perspectives provide a very comprehensive documentation of the example system being researched.

2.3.2 Waterfall methodology

The Waterfall process model approaches software development by defining six key phases. In general, the phases are as follows: Requirements phase, Analysis phase, Design phase, Implementation phase, Post-Implementation phase, Retirement phase. These phases are moved through in a linear fashion. "A critical point regarding the waterfall model is that no phase is compete until the documentation for that phase has been completed and the products of that phase have been approved by the software quality assurance group (Schach, 2001, pg. 49)." In an ideal world, given perfect requirements and perfect understanding of those requirements by the software developers, this methodology may have been more successful. In reality, this is not the case.

There are several problems that consistently arise when using the Waterfall method. The first of these problems presented itself due to the lack of User input throughout the system's life cycle. Typically, the business user would provide input up front without any further involvement until the project has been developed and testing was required for final signoff. Without interaction with the business user during the development process the programmers did not have feedback to assist in correcting false assumptions. Often times, programmer/analysts would build systems based on these initial business requirements and later find that the specifications were incomplete or their assumptions about what the business user wanted were incorrect. The result is loss of precious time and resources that were wasted by developing a system that did not meet a business need. Part or all of the system would then need to be discarded and new resources would need to be allocated to revisit the requirements in hopes of capturing a more accurate picture the second time through this process.

2.3.3 Incremental and Iterative methodology

The Incremental and Iterative method was created with the concept of breaking down projects into a series of smaller Waterfall like iterations. This method allows the delivery of incremental parts of the entire system in the same linear sequence as the Waterfall process previously suggested.

> Each workflow consists of a number of steps, and to carry out that workflow, the steps of the workflow are repeatedly performed until the members of the development team are satisfied that they have an accurate UML model of the software product they want to develop (Schach, 2001, pg. 67).

This is a better approach, allowing the business user to be involved with the development process earlier in the game. This model does a better job at catching the human mistakes that are inevitably made when developing software. It also does a better job at accommodating those applications where the requirements change while the software is being developed. Both the developer and business users knowledge increase with each iteration of the software development and design. As a result, their experience would allow each party to understand the business problem more clearly. This is a definite benefit and improvement over the original methodology providing more opportunities for verifying the software product against the users business needs. Each iteration offers a new opportunity to find inconsistencies and correct them, avoiding costly revisions after the software development has been completed.

2.3.4 Project methodology

While both the Waterfall and Iterative and Incremental Software Development methodologies have many strengths, there was still something missing in these approaches. As systems become larger and more complex, a more dynamic and all encompassing methodology is needed to give greater dimensions into aspects of an organization and the considerations that should also be present during the design of a major system.

Many of the methodologies enforce up front discipline to review the project at hand and provide specific phases or foot prints for the developer to follow to create the final working system. This would fulfill the view from one prospective; however, none of these software development methodologies supports the richness and depth to address the system as a whole entity as effectively as the Zachman Framework. While aspects of some of these methodologies are appropriate for specific components of this system solution, there are many aspects of an application to be considered to capture the system/organization as a whole as well as all of the component parts that need to be considered when developing a system solution. As a result, the Zachman Framework was chosen to provide the blue prints in which all aspects of the example system were viewed. This project shows how the Zachman Framework is a good methodology for small non-Enterprise systems.

One of the advantages of using this framework is that it provides a vast amount of flexibility while providing a comprehensive way to consider the system as a whole. The Zachman Framework is known for creating a structure in which an organization can plug in more specific architectures such that all methodologies work in orchestration with each other creating a comprehensive technology solution. While this is particularly useful when working with a large scale Enterprise Application, the same concepts can be applied to systems of all sizes. The same questions can be asked with respect to the smaller systems needs and this single architecture can be used on a different scale to provide a solid structure, framework and comprehensive approach to systems development.

John Zachman stresses that an organization does not have a single architecture but rather a multitude of different diagrams and documents that comprise the different aspects, perspectives and stages. “An overwhelming rich variety of patterns has been proposed in the literature today

(Jezequel, J M, Train, M. and Mingins, Ch., 2000, pg.14)." Each cell can be expressed independently and reflected by a different architecture or design model. This application of the framework may not use all of the cells, but rather choose those cells that relate directly to the aspect of the system that needs to be captured as critical input into the design process. An implicit assumption will be made for the cells that are not used since all of the cells are necessary for a complete description of the information system/organization.

Zachman's Framework will provide the perfect structure in which to develop the Web based donation system. The appropriate methodology or design pattern will be chosen for the selected cells to demonstrate how these technologies, processes and patterns all fit together creating the blue prints of a final system. This framework has been chooses because it provides complete flexibility and a comprehensive System Architecture needed to develop a system that is able to adapt to the ever changing demands of the Information Technology industry.

2.4 Contribution to the Field

This thesis will prove the scalability of the Zachman Framework. Software developers and analysts will be able to use this work as a demonstration of the comprehensive nature of this methodology and apply it not only to large scale enterprise systems but to projects that are medium to small in size as well. Most of the references to the Zachman Framework are geared toward large, enterprise development application and this thesis will show that its application to non-enterprise applications is equally as successful. By applying the framework to a non-Enterprise Web application, the project will show how this complex framework can be scaled down in such a way that it is appropriate for a project of any size.

Along with showing scalability of the Zachman Framework, this thesis will prove the flexibility of the framework. The Zachman methodology has been proven successful on a large

application and that success can also be carried into projects of a smaller scope. Often times a project scope will be narrow for the initial build but then expand as the company gains more market presence. This thesis will show that the cells of the Zachman Framework can be modified to fit the scope and complexity of the project being developed. The analyst has the ability to decide the level of detail required to capture the information needed for each cell during each phase of its development. The framework simply provides a well organized way to exhibit clear relationships that ensure all aspects of an enterprise are addressed regardless of the order in which each element is established.

This thesis should provide evidence that the Zachman Framework is comprehensive, flexible and extensible for projects of all sizes. As this Framework gains its momentum in the software development arena, this thesis should serve as a point of encouragement for the smaller non-Enterprise application developers that are interested in applying this very robust framework.

Chapter 3: The Zachman Framework methodology

3.1 Zachman's Framework Scaled for Small Systems

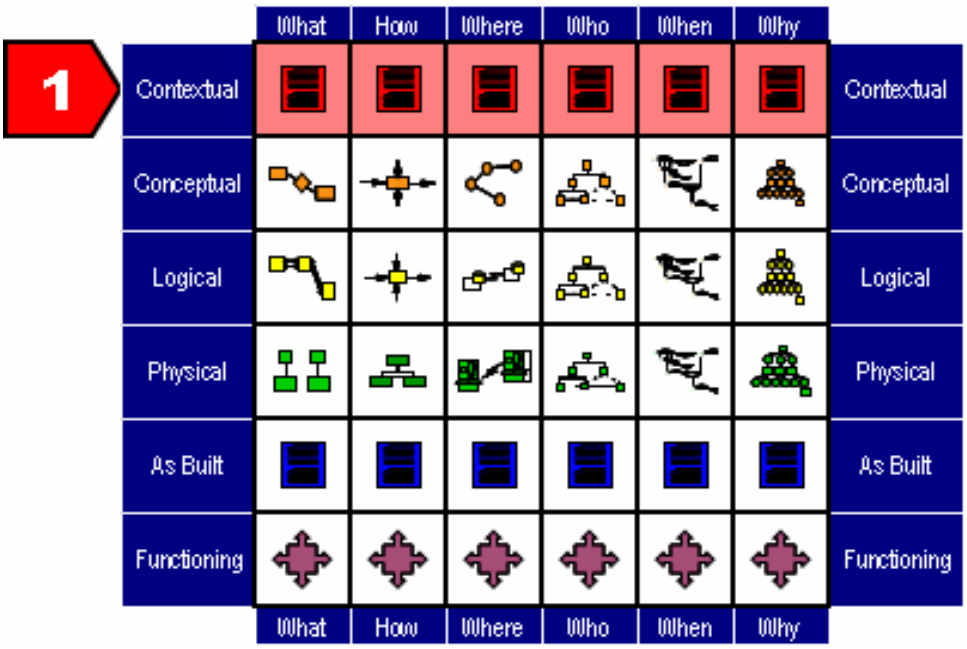
The context of this thesis was to take the existing non-Enterprise system and apply the Zachman Framework as the software development methodology. (See Figure 1.) The size of the example system is small, and relatively independent in nature. It did not need to be integrated into an existing Enterprise Architecture. There are thirty-six cells in Zachman's matrix. The purpose of each row has been presented in relationship to each of the six perspectives described by the framework. This framework was then applied to the example system. Finally, the framework will then be summarized to state its effectiveness in terms of its application to the development of the non-Enterprise System Architecture.

(Figure 1 – Zachman's Framework Matrix)

3.1.1 Row 1: Perspective of the Planner

> The first row of the Zachman Framework describes the prospective of the Planner.
>
> Planning not only identifies the major components of the Information System, but also addresses it's financial viability (costs and benefits), constraints (often imposed internally by legacy systems and externally by the need to connect with other organizations), and scope (what will be part of the Information System and what will not) (Zachman, 2004, pg. 1).

In this case the system is being described from the perspective of the Planner. The Planner would present the business person or persons related to the system who plan and organize the activities of the organization. This would present a high level view of the business needs as they relate to the Data (What), Function (How), Network (Where), People (Who), Time (When) and Motivation (Why). These are all columns in Zachman's Framework and they represent different areas of the system. (See Figure 2)



(Figure 2 – Zachman's Framework – Row 1 - Perspective of the Planner)

3.1.1.1 Data – What is it made of? - Column 1, Row 1. Zachman shows that the "Data" cell in row one begins to describe a list of things that are important to the business. Zachman uses the word "entity" to identify the data aspects of the system. In row one Zachman gives the example of the entity being a class of a business thing (Hay, 1997, pg. 4). Hay (1997) suggests that the data or information shown in the first cell begins to list those things that concern the company and affect its direction and purpose (Hay, pg. 3). This would include any mission critical information that would be needed for the organization or business system to function on a daily basis. This is the essence of what is being captures by this cell.

The Data cell as it relates to the example application will define the high level business information that needs to be captured. In the case of the donation system, the data being captured would be the data related to the donation itself. Since row one is capturing data at a high level, the details of the type of donation and who is making the donation is not explained here. What is important is that the donation information be captured and noted as an important business need.

3.1.1.2 Function – How does it function? - Column 2, Row 1. The second column is concerned with the "How" or "Function" from the Planner's perspective. When describing the function column, Hay (1997) states, "… the Function column describe[s] the process of translating the mission of the enterprise into successively more detailed definitions of its operations" (Hay, pg. 3). In applying this concept to the example system, we begin to identify the high level process or features that the system will need to be able to perform to satisfy the business need. The sole function of the example donation system is to be able to process the donation being made. This would be the starting point regarding the main process needed to solve the business problem.

3.1.1.3 Network – Where are things located? - Column 3, Row 1. Column three identifies the Network and answers the question of "Where" the Planner will be focused in determining system needs. The example system is a web based application so the network is international. The application does process money and the only language used to create the application is English, so the scope of the network is limited to those countries that use U.S. currency and those people who speak the English language. While the system will be hosted by a United States based Internet Service Provider, the application can be accessed internationally. This particular consideration is becoming more and more important as technology expands our ability to access any site located on the World Wide Web. By identifying the Network scope, the requirements for column three, row one are satisfied.

3.1.1.4 People – Who is involved? - Column 4, Row 1. When developing a business system, the scope or boundaries of the system must be defined. "This is simply a list of organizations to which the Enterprise assigns responsibility for work – the 'universe of discourse' relative to people (zifa03, 2003, pg. 4)." The example system is not part of an organization; rather, it will service the need of any individual who wishes to accept Internet based financial donations. So the "People" cell would simply refer to the Planner of the system. Since the example application is small in nature, it did not fit into a larger body of integrated departments. It operates as a stand alone process therefore there was not much elaboration in this area.

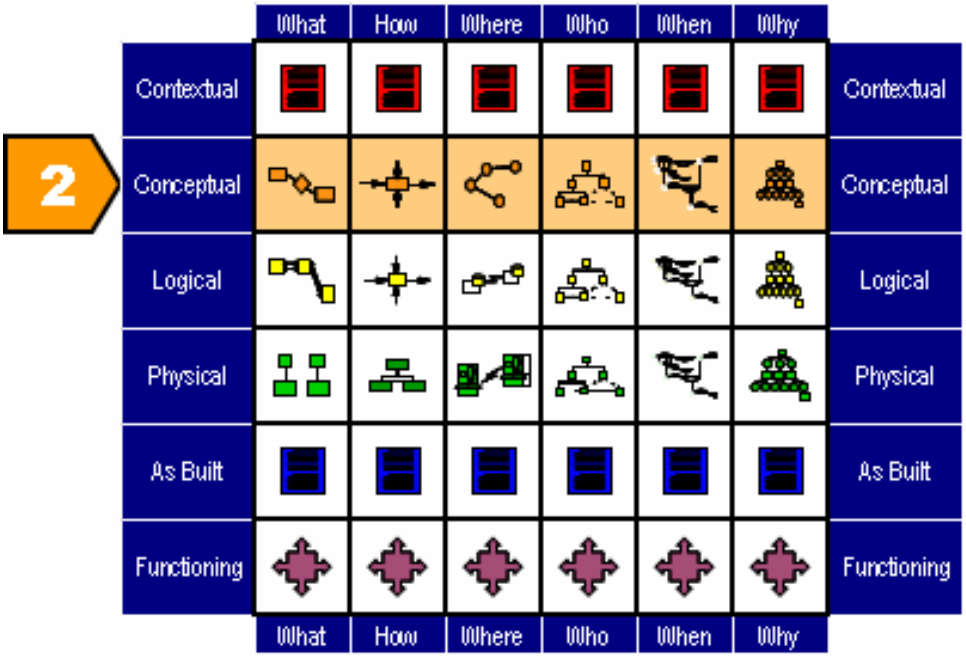
3.1.1.5 Time – When do things happen? - Column 5, Row 1. The "Time" cell answers the question of when things need to happen. This is the focus of Zachman's column five. As referenced in Business Process Trends, time responds to the list of events significant to the business process (Frankel, et al., 2003, pg. 12). From the Planner's perspective, Time is

referring to the business cycle and the events as they relate to the business. With respect to the example system, it is referencing the availability of the overall functionality. Business users want to know when the requested functionality will be available and we will look to this column to provide documentation of this information.

3.1.1.6 Motivation – Why are things done? - Column 6, Row 1. Finally, the "Motivation" cell is considered in column six of the methodology. When Zachman referred to Motivation, this not only answers the question of Why, but it also allows the enterprise to identify its goals and strategies in common language (Hay, 1997, pg. 4). So, again, from a high level, a statement is made about the business as a whole. The Motivation with respect to the example system was to produce a functional Internet based donation system. The statement of the systems motivation captures the overall business need in general language. If there were industry specific vocabulary that could be used to describe or define the system, it would be used as more detailed descriptions are required by the perspectives that are to follow.

3.1.2 Row 2: Perspective of the Owner

Progressing down the matrix of Zachman's Framework, we come to row two. Row two takes on the perspective of the Owner. (See Figure 3) "This [row] describes the models, architectures and descriptions used by the individuals who are the owners of the business process" (Hay, 1997, pg. 3). Ertaul and Sudarsanam (1995) state in their explanation of the Owner's view, "The owner describes this relationship taking into consideration the desires of the users (Ertaul & Sudarsanam, pg. 3)." The Owners view expressed through the different dimensions of the systems development efforts will be presented next.



(Figure 3 – Zachman's Framework – Row 2 – Perspective of the Owner)

3.1.2.1 Data – What is it made of? - Column 1, Row 2. From the business owners point of view we revisit the relationship that the Data has to the system. This particular cell will capture the nature of the data and its relationship to the process. Hay (2000) explains how the information captured by the Data/Owner relationship can be expressed in an Entity Relationship Diagram. This can include relationships such as many to many, n-ary, as well as attributed relationships (Hay). When reviewing the example system and defining the data from the Owners view, an Entity-Relationship Diagram (ERD) would be used to capture this information. The ERD is a data modeling tool that can be used in the analysis process to describe the data requirements needed for the business solution. Three basic elements are captured by using the ERD.

- Entities

- Attributes
- Relationships

Entities are defined. Entities are the things that relate to the system. Attributes are collected. Attributes represent the data as it relates to the entities. Lastly, the relationships between the entities are captured. The author determined that these diagrams would be used to define the different types of tables or databases that would be implemented later on in the Software Development Life Cycle.

3.1.2.2 Function – How does it function? - Column 2, Row 2. Zachman takes a look at the Function cell by answering the question of How from the business owners perspective. Hay (1997) feels that this aspect can be captured by using a Physical Data Flow Diagram (Hay, pg. 3). The definition of a Physical Data Flow Diagram as stated by Davis is "A Physical data flow diagram is a representation of the flow of data into, out of, and between procedures, subsystems or systems" (University of California-Davis, 1996). The physical relationships that make up the example system would be drawn using a Physical Data Flow Diagram. This diagram would graphically depict the relationship between the sponsor, donation system and the merchant service system using lines to indicate the data flows. The author identified the Physical Data Flow Diagram as the diagram that would be added to the archives describing column two, row two of the example system.

3.1.2.3 Network – Where are things located? - Column 3, Row 2. When contemplating the Network needs of the example system, the business owner ponders the question of "Where are things located?" The answer to this question began to define the Network cell as it relates to the system. In row two, column three of the Zachman Framework, the information can be captured by defining the logistics of the Network (Frankel, et al.,2003, pg. 2). When describing the

network logistics, the nodes and links are specified. In the realm of networks, a node represents some sort of device such as a computer, and the links represent how they are connected together. Again, a graphical indication of the network would be constructed based on the needs of the example system. This particular drawing does not have a formal name; however, the diagram would capture the pertinent information related to the networks needed for the business solution.

3.1.2.4 People – Who is involved? - Column 4, Row 2. The Work Flow model is used to represent the People component of the example system. From the Owners perspective, this Work Flow Model Diagram captures how the people are related to the system. "This is the model of the actual Enterprise allocation of responsibilities and specification of work products (Hay, 1997, pg. 4)." By using the Work Flow model, the developer is able to identify the communication with the application owners and the users. This diagram is used to illustrate how the needs of the owners and users will be met by the system. For the example system, a Work Flow Diagram would be developed to capture the relationship of the owner and the users/ people interacting with the system. A user might log on to the system and the owner may request reports from the system. These actions would be captured by the Work Flow Diagram and logged as an artifact to represent column four, row two.

3.1.2.5 Time – When do things happen? - Column 5, Row 2. From the Owners perspective, one of the most important factors is Time. Time, as it relates to the development of the system is represented in column five of John Zachman's' Framework. This cell is responsible for something called the "Master Schedule" (Zachman, 2003). It is a timeline that represents the key deliverable dates and elements of the business solution. "[T]he time column defines when functions are to happen and under what circumstances" (Hay, 1997, pg. 4). When applying this concept to the example system, a time line would be created and saved as an

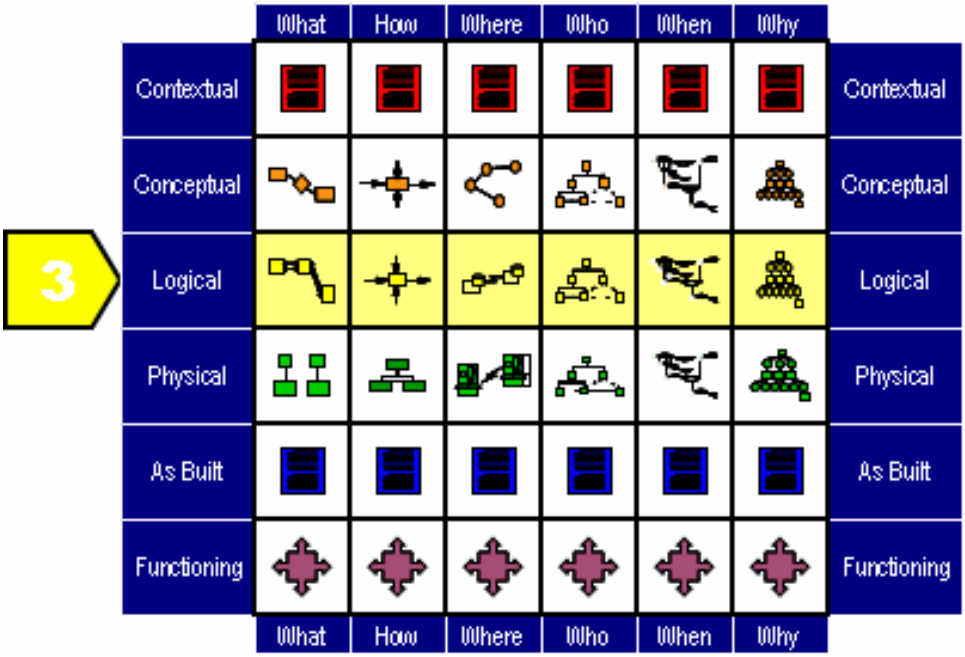
artifact stating those parts of the system that the Owner would be concerned with keeping track of. This would serve as a valuable tracking tool and one that would be used to keep the project on schedule.

3.1.2.6 Motivation – Why are things done? - Column 6, Row 2. The final column in row two is the Motivation column. As stated by an article in Business Process Trends, this column answers the question Why do things happen? (Frankel, et al., 2003, pg. 2). What Zachman captures in this cell is the Business Plan. Business goals and strategies are now translated into specific rules and constraints that apply to an enterprise's operations (Hay, 1997, pg. 4). The Business plan serves as a blueprint and communication tool for the business. The example system would need to be defined in terms of a business plan and this would capture the information needed to complete row two of Zachman's Framework.

3.1.3 Row 3: Perspective of the Designer

Exploring the contents of row three, the Zachman Framework captures the views of the Designer/Architect. These views represent the logical System Model (Department of Veterans Affairs). (See Figure 4) "[Row] three describes those things about which the organization wishes to collect and maintain information, and begins to describe that information" (Hay, 1997, pg. 2). Moving from top to bottom of the matrix that defines this framework, more detail is beginning to be extracted. At the start of this methodology, the information captured was in the form of a high level overview. This overview was explained using general language not specific to the industry. Row three of the framework begins to use more industry specific language. This captures the nuances of the system.

3.1.3.1 Data – What is it made of? - Column 1, Row 3. The Designer can be considered the Architect of the example system. We now look at the first column for row three in Zachman's Framework. This cell represents the Data considerations that need to be made by



(Figure 4 – Zachman's Framework – Row 3 – Perspective of the Designer)

the system Architect. David Hay (1997) represents this information by using a Data model. This model will show the converged entities in a fully normalized fashion (Hay, pg. 3). According to the Wikipedia encyclopedia, "a data model is a model that describes in an abstract way how data is represented in a business organization, an information system or a database management system (Wikipedia)." Using this model, the language that is specific to systems design begins to be used. The example system would be captured using a Data Model showing the data represented by the donation system. This information could be used by the Database Administrator when making considerations about this cell of the example system.

3.1.3.2 Function – How does it function? - Column 2, Row 3. Column two is the Function column. Zachman uses this column to focus on the functions or transformations of the product (Frankel, et al., 2003, pg. 2). Row three begins to model the Information System with respect to the Designer or System Architect's perspective. "[It] portrays [the activities that the enterprise conducts] in terms of data transforming processes, described exclusively in terms of the conversion of input data into output data" (Hay, 1997, pg. 3). Scott Ambler summarizes this cell by showing the application architecture as a means of capturing this information (Ambler, 2006). The application architecture that would be used to develop the example system is called n-tier architecture. The n-tier approach is defined by using the following distinct tiers to group functions in a layered effect:

- Presentation layer
- Business layer
- Service Integration layer
- Persistence layer

These layers in combination with each other define the systems architecture.

3.1.3.3 Network – Where are things located? - Column 3, Row 3. The Network used by the system as seen through the eyes of the Designer/Architect is the focus of row three, column three. "Row three produces the architecture for data distribution, itemizing what information is created where and where it is to be used" (Hay, 1997, pg. 4). This is represented in the Distributed System Architecture. An example of a distributed system is the national telephone switching system or the World Wide Web. The example donation system will be using the World Wide Web as its system network. The donation system would use an open System Architecture so no limits will be place on who can use the system.

3.1.3.4 People – Who is involved? - Column 4, Row 3. Human Interface Architecture is next on the list to be mapped out.

> This is the logical "systems" expression of work flow which would include the specification of the "roles' of the responsible parties including management, administration, knowledge-worker, engineering, marketing, etc. as well as the logical specification for the work products like voice, text, graphics, video, etc. (zifa03, 2003, pg. 1)

The example system was not part of an existing enterprise. There was no hierarchy of usage with respect to who can access different parts of the system. To apply this cell's criteria to the donation system, only two nodes would be shown on the Human Interface Architecture chart. One node would represent a sponsor, that being someone who would like to make an online donation. The other node would represent the System Administrator. The duties of the System Administrator would include requesting reports or maintaining sponsor data. These two nodes together would comprise the diagram symbolizing the Human Interface. This diagram has been identified as the chart that would correspond to column four, row three.

3.1.3.5 Time – When do things happen? - Column 5, Row 3. Column five addresses the impact of Time on the system from the perspective of the Designer/Architect. Hay further explains that row three expresses the events of the business which cause specific data transformations and entity state changes to take place (Hay, 1997, pg. 4). These state changes are modeled using a processing structure. In UML, a State Diagram is used to capture the state changes for the example system. To implement the donation system, a UML State Diagram would be used to capture the relationship of time from the Designers perspective.

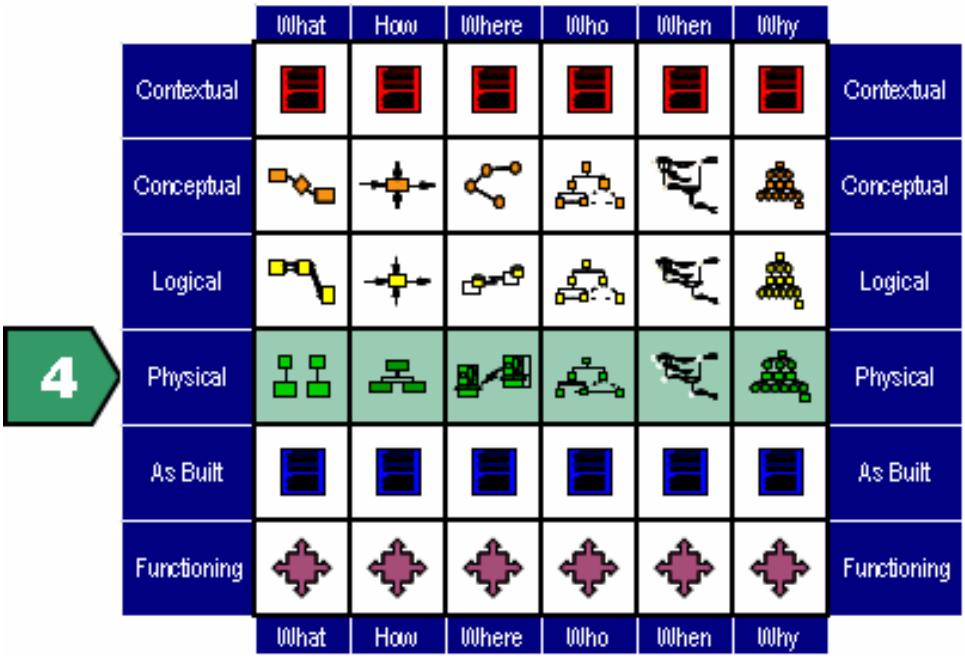
3.1.3.6 Motivation – Why are things done? - Column 6, Row 3. When the Designer of the system wants to document the impact of Motivation, s/he refers to column six of the Zachman Framework. This column intersects with row three to capture the Business Role Model. The business rules are beginning to surface in the capturing of this information. Hay (1997)

summarizes by saying, “business rules may be expressed in terms of information that is and is not permitted to exist. This includes constraints on the creation of rows in a database as well as on the updating of specific values (pg. 4).” When applying the Zachman Framework to the example system, the information related to this cell would show up for the Database Administrator as a column constraint or for the EJB Developer as Business Logic to be executed by the n-tier architecture. The column constraint definition or the business logic would be captured and retained to represent column three, row three. The Zachman Framework does a great job integrating all of the artifacts of the system development process as they relate to the views of the Designer.

3.1.4 Row 4: Perspective of the Builder

The fourth row of Zachman’s Framework represents the Technology Model (Frankel, et al., 2003, pg. 3). (See Figure 5) This captures the view from the Builders perspective. “The builder manages the process of assembling and fabricating the components in the production of the product (Pereira & Sousa, 2004, pg. 2).” The individuals who carry out this work are typically Software Engineers. They use these models and architectures that are produced in response to this row in the framework. The dimensions presented across the horizontal columns will be revisited to describe the different abstractions that impact each perspective. From those perspectives, the system constraints are examined closely and detail will be placed on the final deliverables.

3.1.4.1 Data – What is it made of? - Column 1, Row 4. The Data elements of the example system are examined from the view of the Builder/Developer. To capture the information represented by this cell, the author specifies both a specific design approach and a specific database management system (Hay, 1997, pg. 2). The developer begins to specify the names and



(Figure 5 – Zachman's Framework – Row 4 – Perspective of the Builder)

the columns that will make up the database design for the example application. A relational database management system would be used to persist the data from the donation system. To persist data means to store it in a database table and preserve the information for future use. Oracle would be selected as the Database Management System for the example application used for this thesis. The Oracle database would allow the collection of data to be housed in tables. The Database Management System provides Insert, Update, and Delete to allow manipulation of the data contained in the tables. As we review the impact to the example system at the technology level, we will begin to see more and more details. In this case, the example system would now be committed to using the Oracle features and functionality. While there are several Database Management Systems available, Oracle is the database that would be selected as the back end for the donation system.

3.1.4.2 Function – How does it function? - Column 2, Row 4. To answer the question "How does the system work?" we look to column two of the Zachman Framework matrix. Column two, row four captures the example systems Function as it relates to the business solution. This is where the person responsible for implementing the technology behind this system will begin to sketch out pseudo code (Hay, 1997, pg. 3). Pseudo code is a way that developers take the business rules that they want the logic to carry out and they make their first draft at designing the system. Instead of using the exact syntax of the computer language that they will ultimately use to create this system, they use general wording. For the example system, English like sentences would be produced to capture the story that the actors of the system will be able to carry out. Writing out pseudo code would mark the beginning of the logic that would eventually be translated into a programming language. The logic captured here would be used to represent the Zachman Framework column two, row three.

3.1.4.3 Network – Where are things located? - Column 3, Row 4. The systems Network considerations are next on the list. This is where column three intersects with row four. Zachman captures the view from the lead developer here with respect to the example systems Network. The person responsible for capturing the details of this cell will describe the System Architecture in terms of hardware and software. In attempt to document the answer to the question, "Where are things located?" information is translated into the kinds of computer facilities that are required in each location (Hay, 1997, pg. 4). Since the example system will be developed as a Web application, the specific requirements per location will simply necessitate that any computer wanting to interface with the system must have access to the Internet. As long as the user can open a web browser (i.e. Internet Explorer) and enter the Universal Resource Locator or web address, the donation system will be available for their use. A diagram of the

example system showing the hardware and software requirements would be listed as a deliverable to fulfill this element of the Zachman Framework.

3.1.4.4 People – Who is involved? - Column 4, Row 4. The User Interface is used to show how People will interact with the example the system. The User Interface is described as seen by the Builder. When describing the systems User Interface, the document produced to represent this cell would refer to the web page that is displayed when the web address is supplied in the browser. The user would be presented with a web form that would have text, pictures and buttons describing more about the system and what options the user has in terms of making a donation. The User Interface would show fields that the user could input information into, as well as buttons for the user to click to initiate the next action. The User Interface is the window in which the computer can interact with its operator. In the case of the example system, the User Interface being described is what the User would see on the screen. These screens would be carefully drawn out to provide a very intuitive environment with which the User may interact with. Mock screens would be sketched either by hand or by using an HTML Editor to show the exact image and placements that will be available at run time. A document containing these screen mock ups would be stored as an artifact of the Zachman Framework.

3.1.4.5 Time – When do things happen? - Column 5, Row 4. Column five refers to Time and its impact on the system from the prospective of the Builder/Developer. This component of Zachman's matrix is meant to capture the events and messages that become program triggers (Hay, 1997, pg. 4). Time, in the case of the example system, would be captured as a Sequence Diagram. A Sequence Diagram shows how the user interacts with the system by listing the method or function calls issued by the users actions as well as the function or method calls issued by the system as it responds to the users requests. In a Sequence

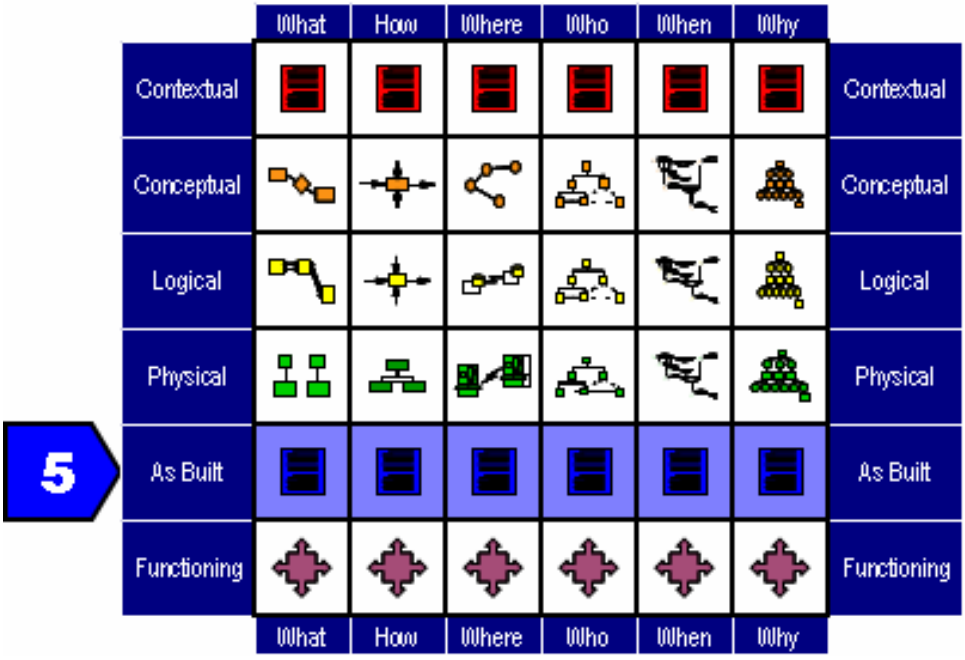
Diagram, the components that make up the system are charted in a linear fashion from left to right. Lines are drawn with arrow heads pointing in the direction of the flow of action. Time is depicted graphically from top to bottom. So, as the user interacts with the system, a line is drawn to show that interaction, and when the system responds, that line is drawn beneath the original one to show both the direction of the request or response as well as the progression of time that has elapsed. This is an example of one way to capture the impact of Time as it relates to the system from the perspective of the Builder.

3.1.4.6 Motivation – Why are things done? - Column 6, Row 4. The last column that is examined is with relationship to the Builders perspective is that of Motivation. Motivation speaks to the purpose and answers the question of why things are done (Frankel, et al., 2003, pg. 4). Here the business rules would be converted to program design elements. For the example system, these elements would be defined in the logic captured by the Enterprise Java Beans. The Enterprise Java Beans would contain the methods needed to carry out the business rules required by the Motivation cell in the Zachman Framework. An example of business rules might be the sequence of events that must take place for the sponsor to make a credit card donation using the web application, authenticate the credit card information supplied, deposit the funds donated into the organizations merchant service account, and notify the user that the transaction has been completed successfully. Business rules can also include activities that happen behind the scenes that include logging the transaction in effort to keep track of transactions occurring through the web application and saving sponsor information to a back end database for future use. As the Builder develops the system, the details of the business logic would be captured in the code that is written. Documenting why things are done or the business logic of the example system is

typical for all software development methodologies. This information would also be captured here in column six, row four of the Zachman Framework.

3.1.5 Row 5: Perspective of the Sub-contractor

Row five describes the view of the Sub-contractor. (See Figure 6) This view will consist of program listings, database specifications, networks all expressed in the terms of the specific programming languages (Hay, 1997, pg. 2). This level would include proper syntax and organization of functions and methods needed to carryout the specific actions as defined



(Figure 6 – Zachman's Framework – Row 5 – Perspective of the Sub-contractor)

in the business requirements document. "These representations illustrate the implementation-specific details of certain system elements: parts that need further clarification before production can begin" (de Villiers, 2003). At this level, all pseudo code is abandoned and hard facts about

the systems needs are captured. As the different dimensions of the systems development effort are explored, the system from the perspective of the Sub-contractor becomes more clear.

3.1.5.1 Data – What is it made of? - Column 1, Row 5. The data column in row five represents the table space needed by the database management system (Hay, 1997, pg. 2). In row five of the Zachman Framework, the details captured begin to reveal the system hardware and software requirements. These are hardware considerations that result from the system and software decisions that were made in the previous rows. In the case of the example system, a Database Administrator would be contracted to determine the amount of table space needed for the donation application. Considerations of the data storage needs would be captured by this cell. Other information would include the data attributes that needed to be retained along with the retention period. Certainly, the Data column information is an important consideration that must be pondered carefully for the business solution to be successful. These considerations would be reflected in the data documentation for row five.

3.1.5.2 Function – How does it function? - Column 2, Row 5. Row five, column two addresses the function of the system from the eyes of the Sub-contractor. This is where the detailed program design would be captured. The programmer converts the detailed program design into source and object code. Developers at this stage have decided what programming language to convert the pseudo code into and they begin to flush out the syntax with the compilers specific for that language. The example system would be developed in Java Enterprise Edition programming language. This language lends itself developing an extensible and robust web application. Having an extensible and robust system is the ultimate goal of using a framework like Zachman's. The resulting document representing this cell would be the compiled program listing.

3.1.5.3 Network – Where are things located? - Column 3, Row 5. The systems Network architecture is a deliverable resulting from the combination of the network column and the detailed representation row (Hay, 1997, pg. 3). The network column from the view of the Sub-contractor is concerned with the communication between computer systems. Such networks involve at least two or more devices capable of being networked together. Row five defines the specific computer types (i.e. Macintosh, IBM, etc.) and the specific protocols (i.e. TCP/IP, SOAP, etc.). The example system would use Internet protocols to communicate across any set of interconnected networks. Internet protocols consist of a suite of communication protocols which include the Transmission Control Protocol (TCP) and the Internet Protocol (IP). The user of the example donation system may have any computer type, so the system is not restricted. Over all, the network aspects of the Sub-contractor are captured in the way that the application would interact with other computers. For the example application TCP/IP would be recorded for this cell of the Zachman Framework.

3.1.5.4 People – Who is involved? - Column 4, Row 5. The People aspect of the example system from the perspective of the Sub-contractor will be identified by the user access permissions (Hay, 1997, pg. 4). These permissions are not only determined for specific tables and columns of data, but they also determine restrictions to specific functions provided by the system. The example system would have two distinct types of users. There would be the sponsor who represents anyone who is interested in making a financial donation. The second type of user would be the system administrator, or anyone who is interested in requesting reports. A report would show donation totals or a list of sponsors and their detailed information. The user would be required to log in to use the donation system and at that time, a determination would be made to distinguish the type of user entering the system. At the program level, the

system would present sponsor functionality to those who are logging in as sponsors, and reporting and maintenance options would be presented to those who are logging in as system administrators. The document created to capture these user permissions would satisfy the requirements for this cell of the Zachman Framework.

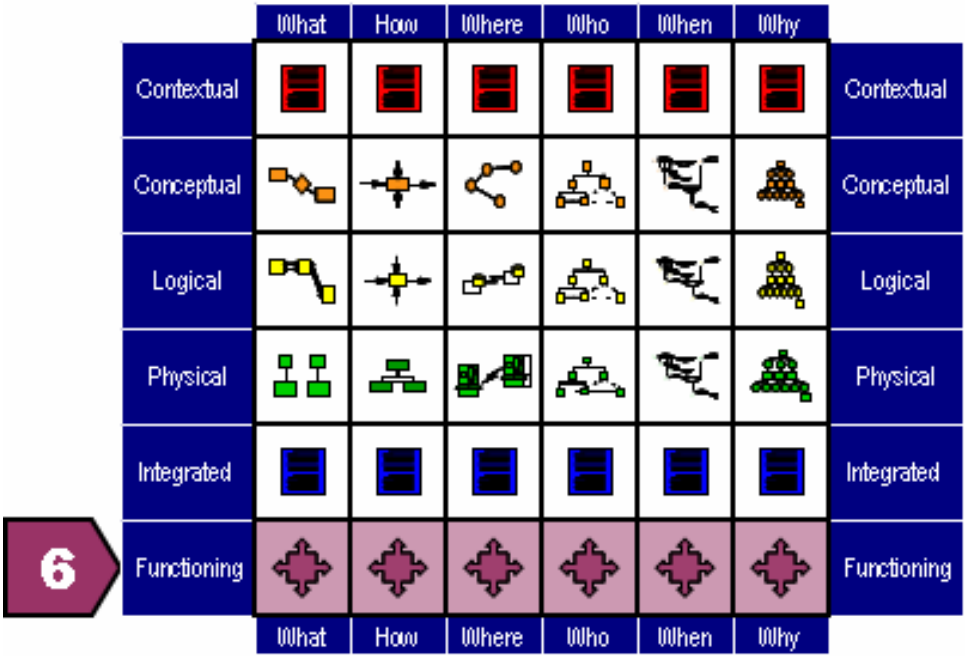
3.1.5.5 Time – When do things happen? - Column 5, Row 5. Time is a critical consideration across the entire development of a business solution (Hay, 1997, pg. 4). In row five, the methods that were identified in row four are developed as actual programs. Where the Sequence Diagrams were previously used, the functionality must be written out in the chosen computer languages, executed and evaluated based on the actual response time needed to present the user with the functionality that was requested. The example system would be developed in Java. So software programs that contained the methods that would be used to respond to the users requests would be developed by the Sub-contractor. These methods would be tested to see if they meet the time constraints defined in previous requirements. System time requirements differ depending on the application, however the end result is to produce a system that responds to the user in a time frame that is appropriate to their needs. The metrics produced by the example system would capture how the system responds to the users' requests. This information would be saved as an artifact of this cell of the Zachman Framework.

3.1.5.6 Motivation – Why are things done? - Column 6, Row 5. The final column for row five holds a place to represent the Motivation of the system. At the level of the Sub-Contractor, the system is held in conformance to the business design. Business analysts should be involved in determining if the programs written by the Sub-Contractor solve the problem initiated by the motivation to build the system. With respect to the example system, at intermediate stages of development, the product would be executed and evaluated to see if it met the business needs.

This process would be done thoroughly as this is the last line of defense before releasing the product as a fully functioning system. So, a use-case document would be put together stating the expectations of the system and the results of the executed test cases. This would be stored to satisfy the requirements of this cell of the Zachman Framework.

3.16 Row 6: Functioning Enterprise

Row six is the final row of Zachman's Framework. (See Figure 7) This row represents



(Figure 7 – Zachman's Framework – Row 6 – Functioning Enterprise)

the functioning system from the perspective of the User. At this level, the system has been tested and has met the system requirements and are now made part of the enterprise. For the Data column, we end up with functioning databases. We implement the many considerations regarding the type of database and how the data is related from the artifacts from the previous rows in the framework. Column two, the Function column, is now represented by the code that

is linked and converted to executable programs. The Network column, fully describes the communication needs of the system. Column four is implemented by training the people who are using the system. Row six deals with time as it relates to the functioning system. Here the business events have been developed such that the system responds to the users requests. Finally, the business rules are enforced by the system and this shows the systems motivation. All of the artifacts created from the previous cells of the Zachman Framework are brought together to form a fully functioning system. The functioning system is what is represented by row six of the Zachman Framework.

3.2 Effectiveness of Scaling the Zachman Framework

"The Zachman Framework was developed taking into consideration all the participants involved in the planning, conception, building, using and maintaining activities of an organization's Information System (Pereira & Sousa, 2004, pg. 2)." This results in a very comprehensible and complete understanding of the requirements needed to build an extensible business solution. A quote made by Loosely summarizes how the Zachman Framework can be used, "It is not a replacement for other programming tools, techniques, or methodologies. Instead, it provides a way of viewing a system from many different perspectives and showing how they are all related (Sowa & Zachman, 1992, pg. 1)". And the conclusion is that the Zachman Framework which has typically been used for Enterprise based application development can also be scaled for use with a smaller, non-Enterprise based applications.

As seen by our example system, each cell of the Zachman Framework can be scaled down to capture the requirements of a small, non-Enterprise web application. While the requirements were not extensive in nature, each requirement was important. The Zachman Framework provided a way to capture the requirements of the example system so that they could be used as a

starting point if an additional implementation with more functionality is planned. These artifacts would document the system so that future implementations could expand on the existing framework adding only the new functionality.

Chapter 4: Project Review

4.1 How the project began

The project began as a result of the author wanting to choose a software development life cycle methodology in her effort to automate a small non-enterprise web based application. After learning about the Zachman Framework, a framework that was primarily used to implement large scale, Enterprise based applications, she began to wonder if this complex framework could be applied to a project that has a smaller, non-enterprise scope.

The web application, www.RunCarlaRun.com, was initially developed to aid in a fund raising effort. The main focus of this web application was to accept donations online and submit a form to the site manager so that further processing of the pending credit card donations could be completed. The target market was small relative to larger enterprise portals. While the application did have web presence, concerns of having multi language compatibility, large development teams and seamless integration with existing enterprise applications were not a concern for this project. While the project was small, it still demanded the detailed requirements gathering process that any well developed system would need for successful implementation, yet other network and global concerns were not a priority for the scope of this project.

The author had learned about the many ways to use design patterns and create diagrams capturing the system hardware and software needs, but the existing software development methodologies were not able to address how those design patterns, object oriented programming techniques and enterprise architecture should fit together.

While having the knowledge of different diagramming techniques provided a way to diagram and capture certain information, the integration of these techniques remained a mystery. After being introduced to the Zachman Framework, the author wanted to find out if the complex framework could be scaled down to fit the analysis needs of a small, non-Enterprise application. The Zachman Framework had been successful in aiding the implementation of Enterprise Applications, but how applicable would it be to a relatively small, non-Enterprise web application? This question is what prompted the research of this thesis.

4.2 How the project was managed

When managing the undertaking of a thesis project, it is important to pace the progress of the project. A timeline was used to manage the thesis project. An outline was created to represent the chapters that would make up the thesis paper. Subsections were added to make sure that major points were emphasized. The chapters and subsections would result in written deliverables. These deliverables were plotted across weekly time intervals to ensure proper pacing of the project. Most software development projects are made up of teams. There are usually business users, business analysts, programmer analysts and developers. Since this project was focused on the analysis using Zachman's framework, the project progression depended on the actions of one person instead of a team of people. Since the schedule was broken down into deliverables that were due each week, it was easy to know whether the project was on track or not.

Managing a project with a small number of people is easy because the communication and coordination efforts are not complicated. Since the author did not have to combine efforts with others in different locations or gather groups of people together to make decisions, the unfolding of the project itself went smoothly. With communication and coordination efforts out

of the way, the most critical tool used here was the time line and staying on track with the proposed deadlines.

4.3 Significant events/milestones in the project

May 12, 2005:	Professional Project Proposal Submitted
June 11, 2005:	Professional Project Proposal Approved
August 1, 2006:	Professional Project Plan Change
September 2, 2006:	Professional Project Rough Draft Submitted
September 17, 2006	First Critical Evaluation of Draft Received
September 27, 2006	Professional Project Second Draft Submitted
September 30, 2006	Received Peer Review on Second Draft
October 8, 2006	Professional Project Third Draft Submitted
October 14, 2006:	Professional Project Completed

4.4 Changes to the Project Plan

Initially the project plan included a build of the example system. The system was going to be fully designed and implemented using Java/J2EE Technology. After discussion about the idea to build, it was recommended that the author focus on the methodology presented by John Zachman and find a way to apply this strategy to the design requirements needed to implement the donation system. Rather than spending time creating the code and implementing the software using a traditional software development life cycle, the work of this thesis was spent researching the Zachman Framework and identifying the supporting diagrams that would be needed to successfully document the requirements for the small non-Enterprise business application. As a result, the example system was not implemented.

4.5 Evolution of whether or not the project met project goals

The project met project goals. The goal of the project was to research the Zachman Framework and determine if it would be scalable for a small, non-Enterprise application. After researching this methodology, a determination was made that the framework is scalable. The author demonstrated this by visiting each cell of Zachman's matrix and determined what information about the example system would be represented within this framework. The criteria used to determine success was to identify if the framework could be applied to the donation system. The author was successful at accomplishing this. Different UML and system diagrams were identified as the graphs and charts that would capture the information outlined in Zachman's methodology. This exercise was done to show how all cells of the framework could be used in a limited fashion to productively capture data needed for the development of the small scale example system.

4.6 Discussion of What Went Right and What Went Wrong in the Project

What went right in the project is the author stayed on track with the proposed time line. Several methodologies were reviewed and selected. The Zachman Framework was evaluated against the non-enterprise system and conclusions were presented regarding how each cell of the framework could be used to capture information about the example system. The project itself had minimal coordination and communication needs. Larger, more complex systems have problems in attempts to coordinate schedules with others and communicate information across teams. Since the project was completed solely by the author, these issues were circumvented and as a result, the project went smoothly.

The author's familiarity with the application contributed to her knowledge of the project scope and current system requirements and constraints. This knowledge played a key role when

evaluating the Zachman Framework against the needs of the example system. Overall, having the project goals charted on a time line, staying on track with the milestones that were set, and having in depth knowledge about the example system were key assets to the success. The project as a whole went smoothly and without any difficulties to note.

4.7 Discussion of Project Variables and their Impact on the Project

The most significant project variable that had an impact on the project was the lack of information on the Zachman Framework being applied to smaller systems. Most of the reference material found referred to large Enterprise systems, with no mention of the smaller to mid size applications. Initially, there was some uncertainty about whether the Zachman Framework was indeed too complex of a model to be used as the software methodology for a smaller, non-Enterprise application. With further investigation and attempts to apply this methodology to the smaller example system, it was discovered that this framework was indeed appropriate for use with a project of any size.

4.8 Findings/ Analysis Results

The findings of researching the Zachman Framework were successful in determining that the framework has the ability to be a scaleable structure. This structure is a perfect software development life cycle for a non-Enterprise small scale system. After digesting the requirements of the Zachman Framework, each cell in the thirty-six frame matrix was applied, on a smaller scale, to the example system. While the framework is typically used for large scale projects that span teams of developers and analysts, the framework can also serve as a useful guide to systems that are engineered by a single developer and serve a smaller set of business users.

Diagrams and tools that are used to capture different views of the system are neatly integrated by using the Zachman Framework. While each of these design documents has value

on their own; the framework serves as a very unique way to see how these diagrams are all inter-related. The views from the six different perspectives are needed regardless of the size of the application being developed. This framework was found to be a very useful tool in guiding the analysis for the non-Enterprise donation system.

4.9 Summary of Results

Zachman's Framework serves as a great methodology for small, non-Enterprise applications. The framework is very flexible and adaptable. If there are cells of the matrix that do not apply to the application being reviewed, those cells can be omitted. The framework can also be amended to include other perspectives that are not covered by Zachman. The analysis process for the example system was given as a helpful road map by using this framework to drive the software development life cycle. The business solution was examined from the highest level down to the implementation of the application code. This gives a very comprehensive view of what will need to take place for a successful implementation of the system. The Zachman Framework proves to be a flexible and extensible methodology suitable for projects of all sizes.

Chapter 5: Lessons Learned and Next Evolution of the Project

5.1 What You Learned From the Project Experience

- The Zachman Framework is Scalable
  - Scale by eliminating row or column data that is not relevant to the project
  - Scale by minimizing the level of detail for any given cell in the matrix
  - Scale by combining the elimination of cells and the minimizing of data captured by a specific cell
- The Zachman Framework provides the means in which to integrate the individual diagramming techniques and charts used to capture the project information
  - UML Diagrams
  - State Diagrams
  - Sequence Diagrams
- The Zachman Framework is robust ensuring the architecture needed to build an extensible software solution
  - Great for developing projects that initially have a small scope but will have a future build to incorporate more functionality
  - Provides documentation for large and small systems
- The Zachman Framework does not have to be used in a linear fashion. Any cell can be evaluated at any given point in time during the project development

5.2 Discussion of Whether or Not the Project Met Initial Project Expectations

The initial project expectations included a full build of the system using one of the more traditional software development methodologies. So, the typical path of Analysis, Design and Implementation would have been followed. A change in project scope occurred when the author

was presented with the concept of applying a non-traditional software methodology. The methodology being evaluated was the Zachman Framework. The quest was to see if the framework could be used to develop a small non-Enterprise web application. The challenge focused on the intricacies of Zachman's Framework and finding out how this framework could be applied to smaller scale projects. At the time, there was some concern regarding if this framework was too robust for a small size system. While the project met the expectations set after the scope change, the project not met the initial project expectations.

5.3 What the Next Stage of Evolution for the Project Would Be If It Continued

The next stage of the evolution for the project would be to complete a set of templates to offer suggestions and guidelines for software developers to use as a resource when creating projects and applying the Zachman Framework.

- Template for Medium to Small Projects – A Template would be created for the typical needs of a Small to Medium size project. The cells from the Zachman Framework would be highlighted or grayed out to show suggestions for optimum project management.
- Special Emphasis Large Projects – A template would be created for the special needs of large project consisting of transcontinental project management. Global considerations would be made when creating these templates to provide a check list for the analyst to optimize the development of large projects.

These Zachman Framework templates would be available in a repository on the internet just as other software development methodology templates are available. This effort would be done in hopes of bridging the learning curve for those who would like to move into a robust software development framework while developing medium to small size projects.

5.4 Conclusions / Recommendations

5.4.1 The Zachman Framework is Scalable

As the project becomes more complex, the first three rows of the Zachman Framework become more important. As shown in Figure 8, the project size is represented along the x-axis and rows of Zachman's Framework are shown from one to six along the y-axis. As the complexity and size of the project gets larger, it becomes increasingly more critical to include rows one through three as part of the project analysis.

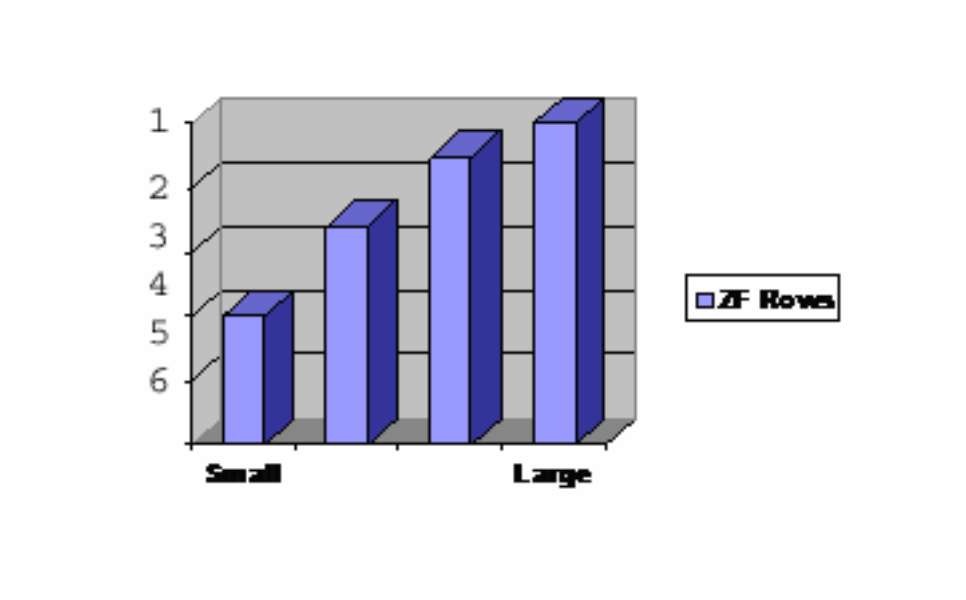


Figure 8 - The Zachman Framework is Scalable

5.4.2 Several ways to scale the Zachman Framework.

- Scale by Elimination – Eliminate the cells of the Zachman Framework that are not appropriate for your project.
- Scale by Limitation – Limit the amount of detail represented by the chart or diagram used to capture the data for any particular cell. Give a high level representation if project does not warrant details.

- Combine Elimination and Limitation – Use a combination of both eliminating cells that are not needed or limiting the detail expressed by the chart or diagram used to capture the data for a particular cell.
- Append Additional Columns or Rows – For projects with greater complexity there may be a need to append additional columns or rows. The Zachman Framework can accommodate these special needs by having the architect add the rows or columns that need to be tracked for a given project.

5.4.3 Conclusions

Zachman's Framework is indeed appropriate to be used as the software development life cycle methodology when developing a small non-Enterprise web based application. Its robust nature lends itself to an application of any size because the framework is flexible. Charts defining the data, network and inter-relations of people and their motivation are orchestrated to work toward the common goal of producing a business solution that meets the needs of the end user. This Framework provides a one stop solution and would be a superior methodology for the architect or engineer of systems both large and small.

References

Alleman, Glen B. (2002). Architecture–Centered ERP Systems in the Manufacturing Domain [Electronic version]. Retrieved September 6, 2006 from www.niwotridge.com/PDFs/ArchCenteredDesign.PDF

Ambler, Scott. (2006). *Enterprise Unified Process (EUP).* Retrieved August 13, 2006, from http://enterpriseunifiedprocess.com.

Deliverable D3.8.1 - Physical Data Flow Diagram - Application Development Methodology (1996). University of California, Davis. Retrieved August 13, 2006, from http://sysdev.ucdavis.edu/webadm/deliverables/d3-/d3-8-1/d3-8-1.htm.

Department of Veterans Affairs. *A Tutorial on the Zachman Framework for Enterprise Architecture.* PowerPoint.

De Villiers, DJ. (2003). Using the Zachman Framework to assess the Rational Unified Process. *IBM.* 4 Retrived August 18, 2006 from http://www-128.ibm.com/developerworks/rational/library/372.html.

Ertaul, Levent, and Sudarsanam, Raadika. (2005). *Security Planning Using Zachman Framework for Enterprises* [Electronic version]. Retrieved August 13, 2006, from http://www.mcs.csuhayward.edu/~lertaul/16_S039EL-S13.pdf.

Frankel, David S., Harmon, Paul, Murkerji, Jishnu, Odell, James, Owen, Martin, Rivett, Pete, et al. (2003). *The Zachman Framework and the OMG's Model Driven Architecture* [Electronic version]. Retrieved August 13, 2006, from http://www.omg.org/mda/mda_files/0903WP_Mapping_MDA_to_Zachman_Framework1.pdf.

Hay, David C. (1997). Zachman Framework: An Introduction [Electronic version]. *The Data Administration Newsletter.* 1 - 5. Retrieved August13, 2006, from http://www.tdan.com/i001fe01.htm.

Hay, David C. (2000). *The Zachman Framework* [Electronic version]. Retrieved August 12, 2006, from http://www.essentialstrategies.com/index.htm.

Hokel, Thomas A. (2006). *The Zachman Framework for Enterprise Architecture: An Overview* [Electronic version]. Retrieved August 13, 2006 from http://www.frameworksoft.com.

Inmon, William H., John A. Zachman, and Jonathan G. Geiger. (1997). *Data Stores, Data Warehousing, and the Zachman Framework: Managing Enterprise Knowledge*. New York: McGraw-Hill.

Jezequel, J M, Train, M. and Mingins, Ch. (2000). *Design Patterns and Contractions*. Boston: Addison-Wesley.

Loosely, C. (1992). Separation and Integration in the Zachman Framework. *Database Newsletter 20,* No. 1, 3-9, Database Research Group, Boston.

Pereira, Carla Marques, and Sousa, Pedro. (2004). Organizational Engineering: A Method to Define an Enterprise Architecture using the Zachman Framework. *Proc. of the 2004 ACM symposium on Applied Computing*. ACM Press.

Schach, Stephen R (2005). *Object-Oriented and Classical Software Engineering*, (6th ed.). New York: McGraw-Hill.

Sowa, John F., and John A. Zachman. (1992) Extending and formalizing the framework for information systems architecture. [Electronic version] *IBM Systems Journal,* Vol. 31, Nos. 3, 1

Wikipedia. *Data Model.* Retrieved July 31, 2006, from http://en.wikipedia.org/wiki/Data_model.

Zachman, J. A. (1999, Reprint 1987). A framework for information systems architecture. [Electronic version] *IBM Systems Journal*, Vol. 38, Nos. 2 & 3, 454 – 470

Zachman, John. (2003). Enterprise Architecture - A Framework. *The Zachman Framework for Enterprise Architecture: A Primer on Enterprise Engineering and Manufacturing.* Retrieved August, 13, 2006, from http://www.zifa.com.

Zachman, John. (2004). *Participant Perspectives in the Zachman Framework. Information Systems Architecture & Organization.* Retrieved July 31, 2006, from http://www.isqa.unomaha.edu/vanvliet/arch/isa/isa-rows.htm.

zifa03. *The Framework for Enterprise Architecture Cell Definitions*. (2003). Retrieved August 13, 2006, from http://www.zifa.com.