Enterprise Architecture in today's economy: no time, no money? No problem!

Richard Freggi, Senior Supply Chain Architect, HP Inc.

***Abstract*— This paper describes the approach and techniques used to accelerate Architecture definition. It focuses on "real world" practical applications of three critical factors:**

- **Engagement approach, including use of the Zachman Framework and the choice of Artifacts**
- **Techniques to speed up Artifact generation using UML (Unified Modeling Language)**
- **Techniques to manage stakeholders and obtain sign-off**

***Index Terms* — Enterprise Architecture, Methodology, Zachman Framework, UML.**

I. INTRODUCTION

Currently several industries are undergoing mergers, acquisitions, splits or reorganizations in an effort to adapt to new business challenges. Enterprise architecture is needed to implement these changes, but paradoxically the investments and time allocated to architecture are being reduced. As budgets tighten, the enterprise architect must learn to "do more with less".

This paper provides some ideas on how to meet these challenges. It is a case study of architecture for a very large and complex domain. Using the approach here described, the time to architecture definition and signoff was reduced from 18 months to 6 months.

Similar approaches can be applied to other engagements, especially where schedule and budget are constrained.

This paper assumes reader knowledge of the Zachman Framework and UML.

II. MOTIVATION AND PURPOSE

Senior management requested definition of architecture to support strategic decisions. The architecture needed to define business process changes, the roles and responsibilities for each organization, list the IT systems required and identify ownership/governance for each system.

The scope included dozens of stakeholders, hundreds of business processes and tens of IT systems worldwide. Peer review estimated that 18 months would be required to generate an architecture, but management required it in 6 months. What follows is a summary of the key factors that were instrumental in achieving this very challenging target.

III. ARCHITECTURE ACCELERATION FACTORS

A. Always do architecture

Previous efforts had to spend time recreating architectural information that was in fact already available but not accessible, due to several reasons: for example the information was in an unusable format, or organized in a way that made it very hard to use, or scattered over so many reports and organizations that it was impractical to collect it.

However in this specific engagement a significant portion of the high level architectural concepts and dependencies could be extracted within days. This information came from engagements that had followed basic architectural methodologies and documented their artifacts in standard formats that were easy to interpret.

B. Start by defining a common language

The most important acceleration factor was reducing the time that stakeholders needed to talk to each other about changes in business processes or IT systems.

Several stakeholders were involved: senior managers, line managers, operational staff, solution vendors, external consultants. Each had different terminologies, different operating models and different views of the company's organization. All these factors slowed down decision making, because stakeholders needed time to understand what was being discussed, analyze it from their own point of view, and then negotiate with each other.

The very first step of this engagement was the development of a domain glossary to be used in process or IT discussions. This dramatically reduced misunderstandings and communication gaps, and decisions could be reached much more quickly.

C. Identify and resolve problems at conceptual level

Problems could be attacked and resolved at any level of detail: from looking at its component parts (physical level) to looking at the 'big picture' (contextual level).

For example, if a business process had unacceptably long cycle time, we could consider in detail each process step, the people involved, the quality of information input, the tools used etc. This would be an example of physical level, and sometimes it was the best approach.

Other times we looked into why we performed this process at all: this would be an example of conceptual level resolution.

It's impossible to generalize, but experience showed that if a problem could be resolved at conceptual level, usually it was easier and faster to do so. The reasons were manifold: simpler data collection, less details to confuse the issue (not seeing the forest because of the trees), less stakeholders involved, better visibility of dependencies with other domains.

D. Leverage conceptual level agreement to drill down to details

Stakeholder agreement on "To-Be" scope, goals, priorities, budget and organization were actively leveraged to speed up problem resolution at more granular levels of detail. For example, specifics of IT systems, business processes and infrastructure could be traced back to the conceptual agreement and evaluated accordingly. Some agreements at conceptual level cascaded down to resolve multiple issues at physical level, providing further acceleration.

IV. OVERVIEW OF ENGAGEMENT APPROACH

The acceleration factors can be described in terms of the Zachman Framework [1],[2]. This provides a useful representation even for engagements using a different Framework.

	DATA *what*	FUNCTION *how*	NETWORK *where*	PEOPLE *who*	TIME *when*	MOTIVATION *why*
CONTEXT *General Manager*	Things important for the business	Business Functions	List of business locations	Organization structure	Business strategy timeline	List of business priorities
CONCEPT *Line Manager*	Semantic data dictionary	Business Process	Business Logistic System	Roles and responsibilities	Business event timeline	Business Plan and Budget
LOGICAL *Architect Project Manager*	Logical Data Model	Application Architecture	Distributed System Architecture	Software interface	Processing timing and sequencing	Business Role Model
PHYSICAL *IT Manager*	Database Schema	Software Specification / Configuration	Hardware and network infrastructure	Presentation Architecture	Control structure	System Role Design
OUT OF CONTEXT *Subcontractor Implementor*	Database Definition Language	Program code	System components	Access and security system	Timing definition	System Role Configuration

John A. Zachman, Zachman International (810) 231-0531

Fig. 1: The Zachman Framework for Enterprise Architecture

Derived from: "Zachman Framework Detailed" by Marcel Douwe Dekker based on earlier work of Phogg2 et al. - self-made, combination of File:Zachman Framework Basics.jpg and File:Zachman Framework.jpg. Licensed under CC BY-SA 3.0 via Commons - https://commons.wikimedia.org/wiki/File:Zachman_Framework_Detailed.jpg#/media/File:Zachman_Framework_Detailed.jpg

A. Typical engagement approaches (non accelerated)

The initial estimate of 18 months duration was based on the typical engagement approach, which began with analysis of the applications (Zachman cell Function - Logical). This felt like a natural place to start, especially for IT teams that understood applications and invested a great deal of time and resources in them.

	DATA *what*	FUNCTION *how*	NETWORK *where*	PEOPLE *who*	TIME *when*	MOTIVATION *why*
CONTEXT *General Manager*	Things important for the business	Business Functions	List of business locations	Organization structure	Business strategy timeline	List of business priorities
CONCEPT *Line Manager*	Semantic data dictionary	Business Process	Business Logistic System	Roles and responsibilities	Business event timeline	Business Plan and Budget
LOGICAL *Architect Project Manager*	Logical Data Model	Application Architecture	Distributed System Architecture	Software interface	Processing timing and sequencing	Business Role Model
PHYSICAL *IT Manager*	Database Schema	Software Specification / Configuration	Hardware and network infrastructure	Presentation Architecture	Control structure	System Role Design
OUT OF CONTEXT *Subcontractor Implementor*	Database Definition Language	Program code	System components	Access and security system	Timing definition	System Role Configuration

John A. Zachman, Zachman International (810) 231-0531

Fig. 2: Non-accelerated engagement approach

The next step was to develop an alignment between business processes and applications.

Database schemas, software configuration, interfaces and infrastructure were based on the application architecture after it had been harmonized with the business processes.

This approach worked well in simpler engagements where the scope was limited, the domain was well understood and the changes to systems and processes were incremental.

However, completing each individual step was difficult and time-consuming in complex engagements with broader scope, more stakeholders and multiple dependencies.

In addition, moving from one step the next also was very challenging, requiring much discussion and back-and-forth analysis/re-analysis. In terms of the Zachman Framework it meant that not only it took longer to define one cell; it also took longer to move from one cell to the next.

Similar problems applied to business-driven engagement that started from the business process (Zachman cell Function - Concept) and then aligned to the application architecture.

Regardless of starting point, the alignment of processes and applications was difficult and slow due to the fact that many business processes used many different applications, and vice versa; the relationship between processes and applications was many:many. The more processes and the more applications, the harder it was to manage.

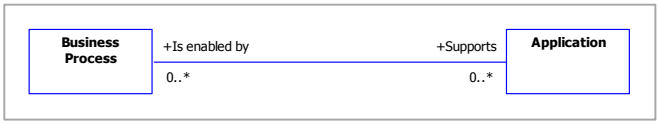

Fig. 3: The relationship between processes and applications was m:m and difficult to manage when multiple instances were involved

B. Accelerated engagement approach

The accelerated approach aimed to move horizontally across the Zachman Framework using the data dictionary to complete each cell quickly; and then to move vertically down in straight lines using each level to complete the next as fast as possible.

	DATA *what*	FUNCTION *how*	NETWORK *where*	PEOPLE *who*	TIME *when*	MOTIVATION *why*
CONTEXT *General Manager*	Things important for the business	Business Functions	List of business locations	Organization structure	Business strategy timeline	List of business priorities
CONCEPT *Line Manager*	Semantic data dictionary	Business Process	Business Logistic System	Roles and responsibilities	Business event timeline	Business Plan and Budget
LOGICAL *Architect Project Manager*	Logical Data Model	Application Architecture	Distributed System Architecture	Software interface	Processing timing and sequencing	Business Role Model
PHYSICAL *IT Manager*	Database Schema	Software Specification / Configuration	Hardware and network infrastructure	Presentation Architecture	Control structure	System Role Design
OUT OF CONTEXT *Subcontractor Implementor*	Database Definition Language	Program code	System components	Access and security system	Timing definition	System Role Configuration

John A. Zachman, Zachman International (810) 231-0531

Fig. 4: Accelerated engagement approach

This is different from the typical approach where many cells are developed by diagonally referring to a single cell (see Fig. 2).

The acceleration factors were applied to this engagement is as follows:

- *Always do Architecture*

Contextual level artifacts developed in earlier engagements (such as business functions, locations, organization, priorities and strategies) were available and could be ported to the engagement scope. This was a significant time saver because work could begin directly from conceptual level.

- *Start by defining a common language*

The starting point was a semantic data dictionary that could be leveraged to quickly define business processes and roles and responsibilities.

The data dictionary not only facilitated conceptual level agreement between stakeholders; it also ensured

	DATA *what*	FUNCTION *how*
CONTEXT *General Manager*	Things important for the business	Business Functions
CONCEPT *Line Manager*	Semantic data dictionary	Business Process
LOGICAL *Architect Project Manager*	Logical Data Model	Application Architecture

Fig. 5: Application architecture was based on mutually consistent process and data

that the process model was fully consistent with the logical data model. Both could be leveraged in the development of the application architecture.

- *Identify and resolve problems at conceptual level*

A particularly long and complex discussion involved system specification for managing orders. It was difficult to reconcile application requirements from different teams.

The solution was to change point of view from the physical (software specification) to conceptual (semantic data dictionary) and review the data dictionary definition of 'order'. It quickly emerged that one team thought of orders as Validated Customer Order / Validated Order Line Item in Fig. 6; while another used the term 'order' to mean 'Load Physical Unit' in Fig. 6, and specifically they wanted to manage Bills of Lading.

The data dictionary helped stakeholders agree that the order management system should not directly associate Bills of Lading with Validated Customer Orders; this was much easier to do in transportation management systems.

This, and many similar cases, shows the value of looking at some problems from the conceptual level to speed up resolution.

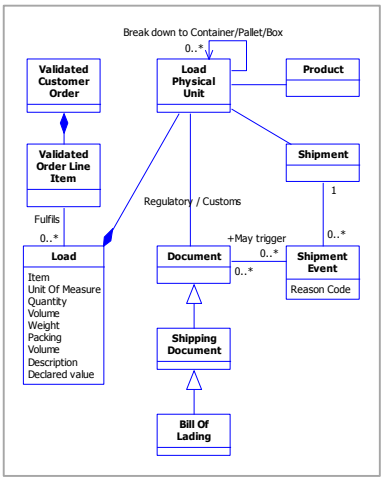

Fig. 6 Conceptual level data dictionary helped to quickly resolve disagreement on physical level software specification

- *Leverage conceptual level agreement to drill down to higher level of detail*

The semantic data dictionary was a great starting point to develop the logical data model. This was important not only to develop database schemas, but also to accelerate the definition of application architecture.

The reason was that the logical data model resolved the many:many relationship between processes and applications, especially because it was consistent with the process model.

This is apparent if we consider that each business process required information as inputs and outputs: processes had one:many relations with the business entities of the logical data model.

Applications provided the inputs and outputs, and could be defined so that each business entity was provided by one application (the so-called 'system of record'). This means that applications had a one:many relation to the business entities.

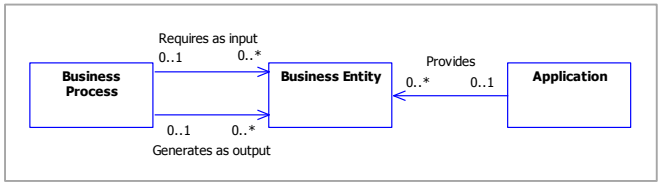

Fig. 7: Business entities from the logical data model resolved the m:m relation between processes and applications

In other words the business entities provided a bridge to map processes to applications, since one and the same logical data model applied equally to both, through a series of one:many relations. These relations remained easy to manage even as the number of processes and applications grew to be large.

In summary, it was much easier to define the applications in terms of what information they managed instead of what processes they supported.

Physical level architecture could be developed quickly because many problems, inconsistencies and contradictions had already been identified and eliminated at conceptual and logical level.

Eliminating as many issues as possible at the conceptual and logical level led to fast progress, because resolving integration issues at the level of database tables, software configuration and infrastructure was difficult, expensive and slow.

V. UML TO ACCELERATE ARTIFACT GENERATION

In this engagement most of the Zachman Framework cells in scope were documented using UML diagrams.

UML is typically used for software engineering at logical and physical level; however several authors have noted that UML is equally useful to describe the enterprise at contextual and conceptual level [3],[4]. Fig. 8 shows a summary of the artifacts and notations used in this specific engagement.

	Data	Function	Newtork	People	Time	Motivation
CONTEXT	Class Diagram	Package Diagram	Package Diagram	Package Diagram	Text	Text
CONCEPT	Class Diagram	Use Case Diagram Sequence Diagram	Component Diagram	Use Case Diagram (Actors)	Text	MS Excel Worksheet
LOGICAL	Class Diagram	Component Diagram	Component Diagram	Sequence Diagram (Objects)	Timing Diagram	MS Excel Worksheet
PHYSICAL	ERD	Component Diagram	Deployment Diagram	Screen shots	Middleware control logic	Configuration parameters and constraints
OUT OF CONTEXT	SQL, DDL, DML, ABAP etc.	ABAP, Java, etc.	Server scripts	Access and security scripts	ABAP, Java, etc	ABAP, Java, etc

John A. Zachman, Zachman International (810)231-0531

Fig. 8: UML artifacts for the accelerated approach

UML provided excellent support for all the acceleration factors described above. Some examples are:

A. *Easy re-use of Classifiers from other engagements*

UML provides a robust and efficient mechanism for Classifier reuse and refactoring, including for example generalization and dependencies.

Accordingly many Classifiers such as Packages, Actors and Classes that had been defined in previous projects could be directly applied to this engagement, especially at contextual and conceptual level.

B. *Fast generation of artifacts for each cell*

UML is well documented and supported by several excellent Open Source CASE tools that are available immediately and without expense [5].

CASE tool features such as Classifier search and documentation (semantics) reduced the time spent collecting and analyzing classifier information.

The CASE tool also automatically generated diagrams and reports, further reduced time spent documenting artifacts. This was much faster than manually drawing, updating and removing inconsistencies in dozens of diagrams using MS Visio or similar drawing tool.

C. *Fast transition from one Zachman Framework cell to the next*

Many Zachman Framework cells shared the same Classifiers: for example Sequence Diagrams and Component Diagrams used Classes and Objects from the Class Diagram. Fig. 8 and 9 show the reuse of Classifiers for the artifacts of this engagement.

This supported quick drill down from conceptual to physical level, but it also worked backwards, i.e. a requirement change at physical level could be quickly propagated vertically to higher Zachman Framework rows and horizontally to other columns, because the impact on each Classifier could be determined as shown in Fig. 9.

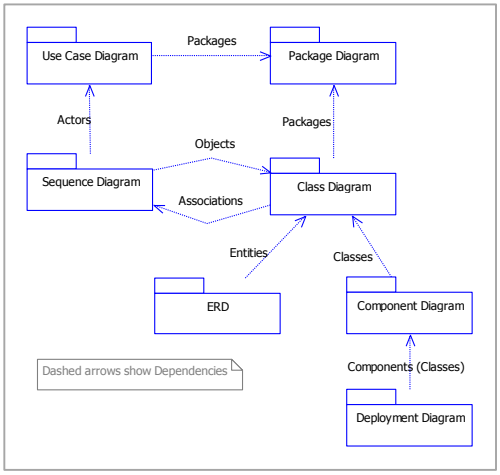

Fig. 9: Re-use of UML Classifiers speeded up generation of Zachman Framework artifacts

This is much harder to do with structured analysis notation such as process flowcharts, IDEF0, data flow diagrams and entity-relationship diagrams.

For example, boxes and arrows in a process flowchart did not correspond directly to entities in the entity relationship diagram, and neither could be mapped into applications and flows of a data flow diagram.

Although the necessary Zachman Framework cells could be documented with these notations, maintaining consistency across cells would have been a major effort.

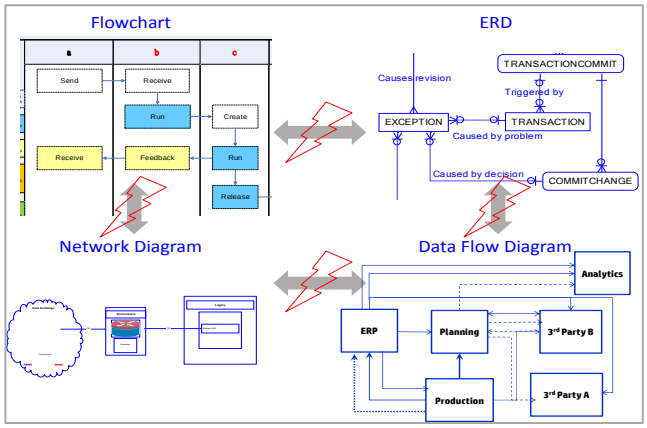

Fig. 10 Structured analysis artifacts did not share elements, therefore they required more effort to generate and maintain

VI. STAKEHOLDER MANAGEMENT AND SIGNOFF

UML artifacts were very effective to communicate key concepts and discuss alternatives, partly because questions could be expressed at conceptual level in terms familiar to decision makers and executives; and also because UML notation could be highly synthetic and helped to filter out non-essential information.

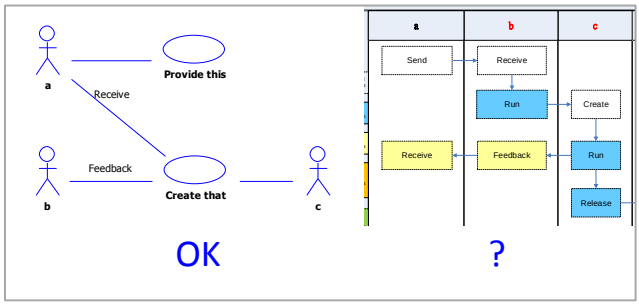

Fig. 11: UML notation (left side) communicated complex questions better than structured analysis notation (right side)

The CASE tool was very useful to identify the Classifiers relevant to the subject matter and quickly generate diagrams and reports that supported managerial decisions.

Artifacts also simplified and accelerated solution fit/gap analysis, negotiation with vendors and project planning. This was a significant factor in stakeholders acceptance and signoff of the architecture, since its feasibility in practical terms was proven.

Artifact	Project Control Document
Class Diagram	Project glossary / definition of terms
	Project scope definition
	Project approach
Use Case Diagrams Sequence Diagrams	Project sponsor assignment
	Project team roster
	Stakeholder management plan
	Test plan definition
	Management of change and training plan
Component Diagram	Project scope definition
Deployment Diagram	RFP, RFQ, vendor Statement Of Work

Fig. 12: Artifacts of the accelerated approach used to define scope, deliverables and governance of implementation projects

VII. CONCLUSION

The accelerated approach described in this case study is based on well-established methodologies and tools. It can be applied to other engagements where speed is a primary concern and can result in increased business value / Return on Investment for Enterprise Architecture.

REFERENCES

[1] John Zachman, http://www.zachman.com
[2] Overview of the Zachman Framework is available at https://en.wikipedia.org/wiki/Zachman_Framework
[3] J. Gorman, *UML For Managers, 2005, ch. 4 page 15*, http://www.codemanship.co.uk/parlezuml/e-books/umlformanagers
[4] Mike Rosen, MDA and the Zachman Framework, 2003, available at http://www.omg.org/news/meetings/workshops/MDA_2003-2_Manual/1-2_Rosen.pdf
[5] Consult Wikipedia for "List of Unified Modeling Language tools", including proprietary and Open Source tools.