What Is Architecture? An Explanation for IT Professionals

Wm. Andrew Wynn. Associate Director, Strategy, Architecture, and Standards
Global Data Warehousing, Bristol-Myers Squibb, Member of the IEEE and ACM
e-mail: andrew.wynn@bms.com

Abstract

This paper addresses the observation that in IT, the term "Architecture" is used widely yet it's meaning is most often vauge and inconsistent. The paper begins with a literature survey of many sources that use the term Architecture. The sources represents the various uses of the term "Architecture". It begins by reviewing Organizational Archtecture and sequentially progress through Enterprise Architecture Planning, Information Systems Architecture and Software Architecture. The second section presents a more specific definition of Systems Architecture then is represented in prior literature survery and highlights how the work Kilov and the RM-ODP complement and extend the methods presented in the more common Architecture definitions.

1. Introduction

The following is a quote from Peter Drucker, which he published in 1945. It is about the value of and necessity of specification and design as part of the system development process. "Automation is a concept of the organization of work…. Automation is not "technical" in character. Like every technology it is primarily a system of concepts, and its technical aspects are results rather than causes. The first concept is a metaphysical one: that there is a basic pattern of stability and predictability behind the seeming flux of phenomena…. Only after these concepts have been thought through can machines and gadgets be fruitfully applied [8]."

2. Literature Survey

2.1 Macro-Level Architecture: Organizational Architecture

In the text Organizational Architecture: Designs for Changing Organizations [22], David Nadler and his associates describe an approach to designing High-Performance Work Systems. The approach is based on the Systems Theory of Organizational Behavior. Thereby, it views any purposeful organization of human beings as a system. In doing so, Organizational Architecture seeks to achieve good "fit" (or congruance), between the various dimensions of an organization. The theory draws on the socio-technical model of work design.

In the text, the authors trace the basis for their approach back to the research done in the United Kingdom in the late 1940's. "Researchers from the Tavistock Institute, studying the introduction of new technology in British coal mines (and later in the weaving industry in India), discovered that technological innovations alone could not explain the differences in performance. In fact, the certain technological changes that were intended to increase performance resulted, instead, in performance declines. Research revealed that high performance resulted when the design of the technical system and the design of the social system of work were congruent. Building on group dynamics and general systems theory, the Tavistock researchers demonstrated that high performance required that the needs of the organization's social system and the needs of the technical system be considered equally and simultaneously in the design process [22]."

The quality of this research and the credibility of David Nadler as a first class academic and management consultant [14] lend signficant weight to the contributions of Kilov [15] and Morabito, Sack and Bhate [20] in their efforts to define an approach to "system" modeling that is coniziant of more than only what is required to move and structure data (which of course is about where most Information Systems Architecture definition efforts stop). For example, consider that IT organizations sometimes fail when attempting process improvement initiatives, such as the SEI/CMM [26, 31, 32]. As Michael West emphasizes, if more consideration were given to the organization as a socio-technical system, before and during such efforts, perhaps they would not fail quite as often [31, 32].

2.2 Mid-Level Architecture: Information Systems Architecture

The Zachman Framework is the most widely known method of defining Information Systems Architecture [33]. Its simplicity makes it a useful communication tool and affords it a good deal of popular appeal. A succinct summation of its strengths and weaknesses is provided by Richard Balicki:

"The framework's strengths include (1) a simple method for classifying design roles and product abstractions; (2) a neutrality with respect to methodologies, tools, and techniques; and (3) a facility for varying levels of abstraction (design roles or "rows") while limiting scope (product abstractions or "columns")."

The framework's weaknesses include (1) its "descriptive" oriented - it serves "to document, rather than enable precise and explicit specifications of, information systems constructs." (2) "the framework does not integrate a product abstraction's design roles. Another way to look at this is the lack of linkage between rows within a column…. (and) there's a lack of linkage between columns within a design role"…. (3) the framework's representations are implementation oriented even at the higher level design roles – it represents "business aspects with implementation oriented constructs (e.g., E-R diagrams, hierarchical diagrams, data-flow diagrams, organizational charts, etc.)" [1].

Enterprise Architecture Planning (EAP), as defined by Steven Spewak [28], is an approach to IT application portfolio planning that is purposefully based on the Zachman Framework for Information Systems Architecture. The approach is a synthesis of the Information Strategy Planning (ISP) stage of the Information Engineering (IE) methodology as defined by James Martin [17] and the Zachman Framework. Spewak's synthesis makes the Zachman Framework useful in way it would not otherwise be. For example, the IE-ISP and EAP methods relate the "What" (Information Needs) with the "How" (Business Functions) from the Zachman Framework as part of defining the "Application Architecture" (in EAP terminology).

Additionally, EAP and IE-ISP both do a great deal to clarify how to organize Zachman's concepts into valuable projects and IT management and administrative procedures.

One publically available set of Enterprise Architecture Planning work products was developed by the United States Department of Defense and is viewable at the C4ISR Architecture Working Group website. (http://www.defenselink.mil/nii/org/cio/i3/AWG_Digital_Library/index.htm).

The Architecture Framework report available at this site is includes a list of deliverables that provide immediate insight into the nature of the approach [5]. A facsimile is provide below:

C4ISR Essential Deliverables by View

All Views

1. Overview and Summary Information: Scope, purpose, intended users, environment depicted
2. Integrated Dictionary: Definitions of all terms used in all products

Operational View

1. High-Level Operational Concept Graphic
2. Operational Node Connectivity Description
3. Operational Information Exchange Matrix
4. Activity Model(s) (supporting deliverable)
5. Logical Data Model (supporting deliverable)

System View

1. System Interface Description
2. Technical View
3. Technical Architecture Profile

Another interesting source of insight about how to gainfully employ the Zachman Framework and EAP is represented in the article "Architecture for a Large Healthcare Information System" [21]. This article describes how the C4ISR Architecture Framework to was used to create an organized operational architecture for the 60+ information subsystems that support the US Military Health System.

2.3 Micro-Level Architecture: Software Architecture

According to Steve McConnell, author of The Software Project Survival Guide [18] and former Editor of the IEEE, Software Journal, a good Software Architecture document describes the following:

1. Overall program organization
2. Ways in which the architecture supports likely changes
3. Components that can be reused from other systems or purchased commercially
4. Design approaches to standard functional areas
5. How the architecture addresses each system requirement

He also provides a Software Architecture Checklist (which is included as an addendum to this paper). McConnell's concerns are very similar in focus to those expressed in the Software Engineering Institute's "Architecture" related publications that I have reviewed

[6, 25, 27]. The concerns are clearly not about Organizational Architecture nor are they about Information Systems Architecture for application portfolio planning purposes as in EAP. Rather, I refer to architecture from this perspective as Micro-Level because it is clearly concerned with "IT System Specification" and "IT System Implementation" (to use Kilov's terms). The term "Micro-Level" is not meant to demean this perspective but rather to simply distinguish it from the other, more abstract viewpoints. Also, I refer to this level as Software Architecture but it is also concerned with System Architecture (or more commonly System Engineering) in that the efficacy of hardware and network components of a computer system application are also considered.

Effective execution of Micro-Level Architecture is critical to the success of higher-level architecture definition efforts. To aid this type of follow-through, McConnell asserts that Architecture is a separable stage of the System Development Life Cycle and it should begin when Requirements are approximately 80% complete. I can certainly see the value of this recommendation, because as an Architect on a large team (of very productive programmers) which does *not* use such an SDLC, I am too often surprised that development efforts have already committed serious architectural "short-cuts" that can not be corrected without missing deadlines. It would be nice to consistently have the opportunity to provide input before its too late.

Lastly, a few of the guiding principles offered by McConnell and others concerned with Software Architecture are priceless. I have included some of the more practical suggestions here (and other more entertaining observations as an addendum).

1. Architectures should be built with a purpose in mind *(This is not as self-evident as it sounds. A lot of people seem to be of the impression that Architecture efforts are not real projects and do not need objectives and a scope. The RM-ODP Enterprise Viewpoint goes a long way toward establishing a structured framework for assuring that objectives, constraints and policies will be explicitly considered.)*
2. Implementation of any architecture vision must be staged *(This should be self-evident, but it never ceases to amaze me, how many of my managers and collueges seem to think that an "Architecture" is something that we can define once for everything – like a type of magic that will simply make everything better.)*
3. "The Best" is the enemy of "The Good" *(I just like this expression. It is best to plan on making compromises rather than being disappointed that you could not reach nirvana via the perfection of your architecture.)*
4. Don't "throw in the kitchen sink" - *(The best architectures are simple and do not look as nasty as the problems they were designed to solve.)*
5. A good architecture limits the number ways in which software components may interface with each other *because software components alone, do not prevent their own inappropriate use !*

Finally, a few misconceptions that McConnell's definition of Architecture clearly debunks include: (1) the idea that Architecture is just about data modeling, (2) the idea that Data and Application Architecture can be separated effectively (more about this later). See the first addendum for a more comprehensive list of software architecture concerns.

3. Toward a More Specific Definition of Architecture

According to Haim Kilov, the definition of system architecture is significantly different from the definition of archicture in general. In an Information Systems context, "we deal with systems that more often than not already exist." Whereas "with the definition from the Oxford English Dictionary (OED), the term "architecture" applies only to something created rather than to something that already exists" [15]. I think this very aptly clarifies one of the most often confused aspects of the role of Information Systems Architecture. It is not simply about development of new systems. It is also about interfacing properly with existing systems and about enhancing and evolving existing systems in an orderly fashion.

According to the International Standards Organization (ISO) Reference Model for Open Distributed Processing (RM-ODP), a system architecture is "a set of rules to define the structure of a system and the interrelationships between its parts [12]." The RM-ODP definition is corroborated by Mario Bunge: "Every system can be analyzed into its composition (or set of parts), environment (or set of objects other than the components and related to these), structure (or set of relations, in particular connections and actions, among the components and these and environmental items) and mechanism (or set of processes peculiar to it, or that make it tick.)" [4].

One the most challenging aspects of Information System Architecture and in fact of Information Management in general, is that things change meaning depending on their context. The challenge is concisely illustrated by Kilov: "The characterization of a particular component as "business," "system," or "technological" is

context-dependent, so that, for example, a component considered "technological" in a business context may be considered "business" in a technological context." This basically says that without careful attention to Viewpoints or Contexts, and definition of synonyms and homonyms, what would be an elegant Information Systems Architecture can readily degenerate in to a "House of Mirrors".

The RM-ODP Defines Five View Points:

1. Enterprise
2. Information
3. Computational
4. Engineering
5. Technology

The RM-ODP Viewpoints provide a thorough array of perspectives for facilitating abstraction and conceptual layering.

This "House of Mirrors" effect combined with the fact that system architecture is not simply about new systems are the two issues that get right to the heart of what is most commonly misunderstood (and consequently mis-managed) about Information Systems Architecture.

To further illustrate the challenge of architecting systems that are equally useful in multiple contexts consider the following two comments: (1) "The real challenge is to design an architecture for a family of (software) products, covering not one but a range of markets, with not just one product in each market but a series of complementary, supplementary, enhanced and eventually replacement products, stretching into the foreseeable future [11]." (2) "…There is no such thing as information in itself: every information flow rides on some concrete (physical, chemical, biological or social) process. Consequently, the definition of the concept of an information system presupposes, at the very least, the concepts of system and process [3]."

Bunge's comment explains what make's Hoare's challenge so challenging. In a word, information is context dependent and therein lies the challenge [2, 13, 15, 30]. Just as Database Management Technology was created to battle the flood of Program Described Files that plauged the infancy of the information age, so too a business domain must be defined precisely to genuinely support the creation of information systems which enable business process integration.

While the logic of this argument is sound, practical challenges face those who attempt it. Consider the following story:

"In 1984, (Ed Yourdon) consulted, during a two and one-half year period, on the practical application of Real-Time Structured Analysis at a major aerospace company. His observations were interesting and yet disturbing. One team of analysts he studied (the "DFD Team") started their projects using data flow diagrams to develop an overall functional decomposition as a framework for further specification. Meanwhile, a second team of analysts (the "Data Base Team") started by focusing on the information the system needed to do its job and then building an information model (also known as an Entity-Relationship Diagram or a Semantic Data Model). Over time, the DFD Team continued to struggle with the basic problem of space understanding (e.g., the details of what happens when one controller hands off responsibility for an aircraft to another controller). In contrast, the Data Base Team gained a strong, in-depth understanding of the air traffic control. Yet the results did not mesh together; worse, they contradicted each other. In principle, these two models should somehow come together. Yet under the pressure of schedule and budget, both products moved unresolved into preliminary design, with the hope of resolving the discrepancies at that time. Sadly, the Data Base Team was perceived as irksome, even somewhat as troublemakers; people (and their careers) paid the price for this major rift and its untidy resolution.

In 1987 and 1988, (Ed Yourdon) saw this same pattern develop on projects at a federal government agency and a state government agency. The DFD Team marched on ahead in time and political power. The Data Base Team gained tremendous insight, vital to analysis but all too-often ignored. And again, the Data Base Teams and their leaders were perceived as troublemakers. Repeatedly, in practice, separate notations and strategies for different process and data models have kept the two forever apart." [7].

In my own, professional experience I have seen this nightmare played out again and again. Coad and Yourdon felt the answer was an Object-Oriented Analysis methodology that integrated process and data model notations. Kilov has defined a modeling and notation methodology that achieves a greater degree of notation uniformity (and conceptual integrity) than anything else I have studied [15]. If the problem Coad and Yourdon so clearly illustrated can be solved with a notation method, than our profession is certainly on the verge of resolving this issue. However, by my estimate, I do not believe our profession will heal the schism between the "DFD" crowd and the "Data Base" crowd until we collectively learn to vastly improve our management practices and our collective level of professional knowledge. I can not help but think that best practices frameworks such as the SEI/CMMI and Luftman's Strategic Alignment Model (SAM) [16] must be an integral part of the solution.

Another part of the solution must improved and more thorough undergraduate education for Information

Systems Professionals. I think a 5 year program as is par for the course in Engineering and Architectural studies is long overdue. I also think a 3 year professional degree in Information System similar to a Law degree or an MBA is warranted. Furthermore, I am very much in favor of informational systems professional licensure rather than volutary certification. If this sounds harsh consider that the Standish Group statistics also sound harsh. Maybe not quite as harsh as they did twenty years ago, but does anyone think higher professional standards would worsen those statistics?

Also, I cannot help but think that simply measuring our own productivity and quality should come before, during and after attempting to effectively change anything significant about our culture and work processes [10, 23].

All of the architecture methods referenced in section one of this paper have relatively loose definitions of their components and component relationships. Kilov's semantic modeling techniques have great potential for specifying and implementing the concepts of every type of architecture presented in this paper. Kilov's philosophy is strong in exactly those places where other approaches to architecture tend to be weak. The following diagram is from the "Traceability" section of Kilov's Business Models text [15].

Kilov's Information Management Project Structure

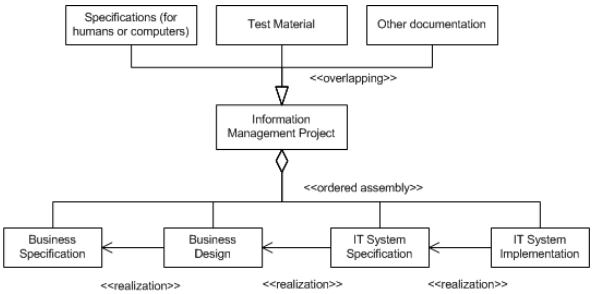

Figure 1: IM Project Structure

Richard Balicki aptly likens Kilov's Information System Components (or "areas of concern") to Zachman's Design Roles.

Kilov's Four Areas of Concern

1. Business Specification
2. Business Design
3. IT System Specification
4. IT System Implementation
5.

Zachman's Design Roles

1. Planner and Owner
2. Designer
3. Builder
4. Sub-Contractor

As such, these areas of concern are somewhat analigous to view points as defined in the RM-ODP. They are also somewhat analigous to the system development life-cycle stages and the workflow streams of the Rational Unified Process.

As the reader will recall the Zachman Framework is particularly weak with respect to the relationships between view points (rows) and domains (columns). In his discussion of Traceablity Kilov addresses this problem with the following specification of the <<realization>> relationships between each of his view points in the diagram above.

Kilov's "Realization" Relationship Specification

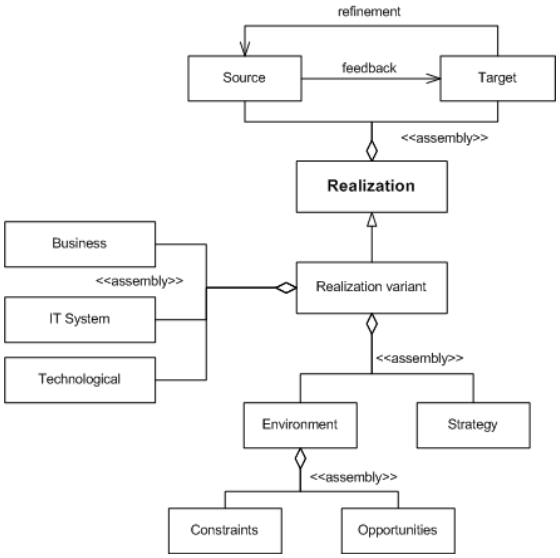

Figure 2: <<realization>> specification

I will resist the temptation to insert a full quotation of Kilov's explanation of the above two diagrams (and encourage the reader to read the book instead). Rather I will simply point out what I find most significant about these diagrams in light of the focus of this paper.

1. According to Kilov's analysis of the structure of an Information Systems Project, the activities that are often vaugely referred to "architecture" of on kind or another, Kilov more precisely refers to as Business and System Specification and Design activities. This directly addresses my primary concerns of (1) the need to be clear about what we mean when we use the term architecture and (2) the need to be clear

about deliverables of architecture activities. In Kilov's analysis, the deliverables are I think very appropriately described as specifications and designs.

2. According to Kilov's analysis, the structure of the Realization relationship is that in each instance it is composed of a strategy, constraints and opportunities and some appropriate combination of business, technological and / or IT System components. The significanct aspect of this from the perspective of our current Architecture discussion is that Kilov's definition is thorough and flexible enough to specify the work-products and the relationships between the work-products of architecture at any level of abstraction. Kilov's Realization relationship is exactly what is missing from the Zachman Framework and is therefore an invaluable complement to it.
3. Kilov includes feedback and refinement in his Realization specification because information system architecture and specification is not strictly a "top-down" affair. Which speaks directly to one of the important and unique characteristics of system architecture we discussed earlier – it is not only concerned with creating new systems but is almost always also concerned with integration with existing systems [15].

One final thought about something interesting I recently came across in my reading. In a real project organization documented in the Project Management Journal [19], there are *many* teams that are "architecture" teams, including the Data Architecture Team, the System Architectre Team and the Software Architecture Team - all on one project team. This is reflective of a thought I have often had about what is required to consistently attain high standards in all aspects of software development – good architecture is *everyone's* job !

References

1. Balicki, R., "Synergizing Zachman's Architecture Framework with Kilov's Information Modeling", Proceeding from the Ninth OOPSLA Workshop on Behavioral Semantics, OPSLA 2000, Northeastern University, College of Computer Science, (October 15, 2000).
2. Berners-Lee, T., Hendler, J., and Lassila, O., "The Semantic Web", Scientific American Special Online Issue, Scientific American, (April, 2002).
3. Bunge, M., (Reply to Mattessich on the Foundations of the Management and Information Sciences. In: Studies on Mario Bunge's Treatise. (Ed. by Paul Wiengartner and Georg J. W. Dorn). Amsterdam-Atlanta, 1990, pp. 641-642.)
4. Bunge, M., Philosophy in Crisis, Prometheus Books, (January 2001).
5. C^4ISR, Architectures Working Group, C^4ISR Architecture Framework, Version 2.0, United States of America, Department of Defense, (December 18, 1997).
6. Clements, P. and Northrop, L., "Software Architecture: An Executive Overview", Technical Report, CMU/SEI-96-TR-003, ESC-TR-96-003, SEI Website, (February, 1996).
7. Coad, P. and Yourdon, E., Object-Oriented Analysis, Yourdon Press Computing Series, Prentice-Hall, (1990).
8. Drucker, P., The Practice of Management, Chapter 3: The Challenge to Management, Section: What is Automation?, Harper & Row, (1954).
9. GAO, United States General Accounting Office, "Information Technology Enterprise Architecture Use across the Federal Government Can Be Improved", A Report to Congressional Committes, US GAO, (February, 2002).
10. Goldenson, D., et. al., "Measurement and Analysis in Software Process Improvement", Chapter 37 in IT Measurement, International Function Point Users Group, Addison-Wesley, (2003).
11. Hoare, C. A. R., "Software: Barrier or Frontier", Oxford University Computing Laboratory (November 23, 1999).
12. International Standards Organization (ISO), Reference Model – Open Distributed Processing 2, Architecture, (1998).
13. Kent, W., Data and Reality, Elsevier Science Ltd; 5th repr. 1990 edition (January 1, 1984).
14. Kerns, D. and Nadler, D., Prophets in the Dark: How Xerox Reinvented Itself and Beat Back the Japanese, Harper Collins, (1992).
15. Kilov, H., Business Models: A Guide for Business and IT, Prentice-Hall, (2002).
16. Luftman, J., Managing the Information Technology Resource: Leadership in the Information Age, Prentice Hall; 1st edition (April 28, 2003).
17. Martin, J., Information Engineering, Book II, Planning and Analysis, Prentice-Hall, (1990).
18. McConnell, S., The Software Project Survival Guide, Microsoft Press, (1998).
19. Metzger, J., "JWARS: A Case Study", The Project Management Journal, Project Management Institute, Volume 34, Number 4, (December 2003).
20. Morabito, J., Sack, I., and Bhate, A., Organizational Modeling: Innovative Architectures for the 21st CenturyPrentice-Hall, (1999).
21. Mukherji, R., et. al., "Architecture for a Large Healthcare Information System", IT Pro, IEEE Publications, November | December, (2002).
22. Nadler, D., et al, Organizational Architecture: Designs for Changing Organizations, Jossey Bass, (1992).
23. Putnam, L. and Myers, W., Five Core Metrics: The Intelligence Behind Successful Software Management, Dorset House, 2003.
24. Rood, M., "Enterprise Architecture: Definition, Content and Utlity", document 0-8186-5705-7/94, IEEE (1994).
25. Shaw, M., and Clements, P., "A Field Guide to Boxology: Preliminary Classification of Architectural Styles for Software Systems", Software Engineering Institute, (April, 1996).
26. Software Engineering Institute (SEI), Process Maturity Profile, SW-CMM, (September, 2003).
27. Software Engineering Institute (SEI), "Essays on Software Architecture", SEI Website (www.sei.cmu.edu/architecture/essays.html), (2004).
28. Spewak, S., Enterprise Architecture Planning: Developing a Blueprint for Data, Applications and Technology, John Wiley & Sons, (September 1993).
29. Sprung, C., "Lesson Learned About Architecture: Unstructured and Eclectic", retieved from the SEI website, (www.sei.cmu.edu/architecture/essays.html), (2004).

30. Strong, D., Lee, Y., and Wang, R., "Data Quality in Context", Communications of the ACM, vol. 40, no. 5, Association of Computing Machinery, (May, 1997).

31. West, M., "Applying Systems Thinking to Process Improvement", CrossTalk: Journal of Defense Software Engineering, United States of America, Department of Defense, March, 2004.

32. West, M., Real Process Improvement Using the CMMI, Auerbach Publishing, 2004.

33. Zachman, J., "A Framework for Information System Architecture", IBM Systems Journal, vol. 26, no. 3, (1987). IBM Publication G321-5298.