The Four-Domain Architecture: An approach to support enterprise architecture design

by B. Iyer
R. Gottlieb

Managing an enterprise architecture is a challenging task. While careful planning typically goes into its design, an enterprise architecture actually emerges as a result of implementing individual projects. It is this de facto architecture, not the conceptual one, that provides the capabilities for executing business strategies, and understanding this emergent architecture is of paramount importance. In this paper, we present the Four-Domain Architecture (FDA), which integrates business process, information, knowledge, and elements pertaining to infrastructure and organization. The FDA approach can help guide the development of both the conceptual and emergent architecture. The FDA helps an enterprise in the definition, design, and creation of a set of tools and methods to support frameworks such as the Zachman framework.

As an enterprise grows in size and complexity, several factors impede its abilities to solve the problems that it faces. The point is rapidly reached where the factors that come into play in structuring and conducting the business of the enterprise become too numerous and complex to manage. When working on such complex systems, designers have typically dealt with this complexity by breaking them into subsets or domains that are less complex than the original system.[1–3]

In the case of information systems, the abstraction used to deal with complexity is called an *architecture.* An architecture is a system design that specifies how the overall functionality of the design is to be decomposed into individual functional components and the way in which these components are to interact to provide the overall functionality of the system. The decomposition of the enterprise into manageable parts, the definition of those parts, and the orchestration of the interaction among those parts constitutes the *enterprise architecture.* The orchestration of the interaction is governed by a set of design rules and principles, also called the organization's knowledge architecture.[4,5]

Architecture design has benefited from the work done in the context of systems architecting[6] and the design of buildings[7] and information systems.[8,9] Other authors writing on product design and development[10,11] have stressed the importance of managing the evolution and renewal of product architecture for sustained competitiveness. In that context, the architecture of a product refers to its overall design concept, which serves both as a basis for product innovation and as a constraint on the variety of product versions. In the context of this paper, we focus on a set of components that enable the flexible retooling of the enterprise and the creation of supporting environments for different business contexts. As such, our enterprise architecture, the Four-Domain Architecture (FDA), reflects an integration of business processes, engines, data sources (e.g., databases

and knowledge bases), visualization tools, dialog managers, infrastructure, and organizational resources.

In presenting the FDA, we separate the business of creating an enterprise architecture (i.e., the processes of defining and building the models of the enterprise and the requisite organizational resources) from the business of doing the work of the enterprise (the building and selling of products and services, the running of the enterprise itself, and the organizational resources engaged in this work). The former we call "Architecture-in-Design" (AID), and the latter "Architecture-in-Operation" (AIO). AIO provides the content for the enterprise's models.[12] Consequently, we split the FDA into two "cross-sections" or subarchitectures, the AID and the AIO.

To provide an enterprise with some guidance for making decisions about information technology (IT) projects, planning sessions are held to define the ideal architecture. The AID thus defined is expected to provide guidelines for project-level decision making. In reality, however, these guidelines, even when available, are rarely followed. A firm's architecture—its AIO—emerges as a result of each project's impact on applications, data, and technology. Therefore, it is the AIO that becomes the de facto enterprise architecture, and consequently, it becomes the architecture most in need of management.

The Four-Domain Architecture

The concept of an IT architecture does not have a universally accepted definition in either the research or industry context.[13] Although at a very high level, we can describe the IT architecture as a capability to integrate technical components to meet business needs, in reality the concept is much more complicated. Previous studies have concluded that an IT architecture includes a group of shared, tangible IT resources that provide a foundation to enable present and future business applications.[14–17]

There are many elements within the information systems world with which to build an architecture: networks, computers, terminals, programs, cabling, data sources, procedures, and so forth.[18] By grouping like elements into domains, domain-specific architectures can be constructed which reflect a common composition and are simple and clearly focused. To be efficient, the elements chosen should have the following characteristics:

- A reasonable level of abstraction (too high a level of abstraction would mystify implementors, while too low a level would result in a welter of "nuts and bolts"),
- Adequate coverage of the real world (i.e., the architecture needs to be complete, without "holes" indicative of situations not planned for),
- Reasonably familiar and accessible concepts,
- A basis in significant philosophical and practical experience, and
- Entities in one domain that differ from those in other domains, while maintaining a similarity to each other.

FDA divides the enterprise into four domains and tailors an architecture model for each. The domains are independent and therefore can be driven by the external factors for which each is designed. Each domain, as described in the following subsections, encompasses a traditional area of expertise, which provides a unifying discipline for each model.

The four domains

The four domains of the FDA, and some examples of elements of the domains, are shown in Table 1.

Process domain. This domain includes the processes, procedures, business tools, tasks that encode business rules, and dependencies required to support the various functions within a business. It includes the applications needed at the levels of operations, management control, and strategic planning. For example, this domain may include recurrent tasks or routines[19,20] that encode design rules and principles. These may come in the form of communication channels, both formal and informal, or in the form of information filters and strategies.[21]

Information/knowledge domain. This domain includes business rules and business data and information of all types, their usage, interrelationships and demographics, as well as their definitions, ownership, distribution, and composition. Meta-data, system data, and operational data are also included within this domain.

Infrastructure domain. This domain includes hardware and facilities, system software, data storage resources, networks and communications, human interfaces, and other underlying technologies. It is the platform that supports the activities and interfaces of the other domains.

Organization domain. This domain includes business people and their roles and responsibilities, or-

Table 1 Typical elements of the Four-Domain Architecture

Process Domain	Information/Knowledge Domain	Infrastructure Domain	Organization Domain
Business context engines	Business data	Computers	People
Planning engine	Business profiles	Operating systems	Roles
Visualization engine	Business models	Display devices	Organizational structures
Business tools	Data models	Networks	Alliances

ganizational structures and boundaries, as well as their interrelationships to alliances, partnerships, customers, suppliers, and other stakeholders in the enterprise.

In our view, the FDA as described above can support frameworks such as the one proposed by Zachman.[9]

The Zachman Framework

In 1987, John Zachman introduced his framework for the definition of the architecture of information systems.[9,22] The concept was inspired by the millennial disciplines of classical architecture and the more recent development of the disciplines and methods successfully adopted for the creation, design, and production of complex machine systems such as airplanes. The construction of information systems (and, more recently, enterprises) exhibits complexities and characteristics no less daunting and challenging than the creation of large edifices or machines. Therefore, a great deal can be learned by observing how the expert practitioners of those arts and sciences go about their work. The craft of designing and building systems based on computer technology is rendered more difficult by the general proliferation of that technology, its wide dissemination throughout the enterprise, and its rapid evolution.

Unfortunately, discontent with the general quality and usefulness of information systems has not resulted in a groundswell of recognition of its causes. Unlike design failures of buildings or airplanes (e.g., collapsing atria, tail assemblies that fall off), architectural design failures of information systems have not caught the public's attention or its wrath. Indeed, two recent events, the Y2K concerns and the Mars Lander debacle, have already faded from view without generating any visible heat, much less demand for removing the root causes of these events.

The Zachman framework (referred to hereafter as "the framework") for dealing with the weaknesses of the state of the art of information systems architecture was prescient. It explicitly recognized the stylized roles played by key actors (e.g., owners and users) in the creation of buildings and aircraft, how they are involved in the related processes, and their unique informational needs and contributions. In addition, it posited a series of aspects or views of interest to each actor, and an appropriate (and unique) representation for each instance. The resulting aspect matrix, originally five by three and subsequently expanded to a five-by-six matrix, presented an exhaustive set of definitional, design, and implementation artifacts for the complete and coherent ideation and construction of any complex system, and, specifically, information systems. The six columns of the matrix can be tied to the basic set of interrogative primitives[22]: "What, how, where, who, when and why." In the framework, these translate to *data, function, network, people, time,* and *motivation* as shown in Figure 1. The aspect or view of the whole that is provided by each column is unique in its nature. Within each column, however, each row represents the needs (identical in nature) of different stakeholders and their concerns. The rows indicate the architecture's scope, enterprise model, system model, technology model, and components.[23]

With the passage of time, it was realized that limiting the use of the framework to the information systems of an enterprise was not justified, given the background and thinking that led to its creation. Because the distinction between an enterprise and the processes, data, and infrastructure of which it is composed is purely arbitrary, the framework focused instead on the abstraction of the enterprise architecture.

The Zachman framework is not an architecture and was never proposed as such. It can be thought of as a multidimensional visual checklist. The artifacts that instantiate the cells of the framework for a given architecture are indeed architectures or subsets of an architecture, depending on one's point of view. The framework itself does not impose architectural rigor,

Figure 1 Six-column Zachman framework (With the exception of the first row, the items in the cells are examples.)

	Data	Function	Network	People	Time	Motivation
Scope	List of things important to the business	List of processes the business performs	List of locations in which the business operates	List of organizations/agents important to the business	List of events significant to the business	List of business goals/strategy
Enterprise Model	Entity/relation diagram	Process flow diagram	Logistics network	Organization chart	Master schedule	Business plan
System Model	Data model	Data flow diagram	Distributed system architecture	Human interface architecture	Processing structure	Knowledge architecture
Technology Model	Data design	Structure chart	System architecture	Human/techology interface	Control structure	Knowledge design
Components	Data definition description	Program	Network architecture	Security architecture	Timing definition	Knowledge definition
Functioning System	Data	Function	Network	Organization	Schedule	Strategy

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

although following its constructs would reduce the probability of mishaps in an architecture defined using the framework. In fact, in later work, Zachman introduced some principles[22] to guide the use of his framework and also speculated about some of the tools and methods that would make the framework's adoption less labor-intensive.

These considerations underscore and motivate this paper. To design, define, and build a tool set and methods to support the framework requires the use of the framework itself, in order to help achieve architectural integrity. This recursive requirement, in effect, requires an architecture for its realization. This paper proposes such an architecture (the FDA) and describes how this architecture would be used to define the tools and methods needed to instantiate the various cells of the framework. The paper demonstrates the use of the framework in the proposed architecture and shows how the framework could be created by using the proposed architecture.

The framework and the FDA: A general approach

The cells of the framework represent information which can be thought of as a set of nested tree structures. For example, if we think of the set of systems that represents a complex product family, a tree would show how the various individual products and options are related, how they interrelate to create the various "sellable" instances of the products, and how those instances are placed within the family. The product family, so constituted, could be mapped against a variety of external realities (markets, product requirements, competitor offerings, etc.) to show coverage and completeness of the product family.

The next lower level would decompose the products into assemblies, and the assemblies could then be broken down into components. The process for doing this is recursive, and a single engine could be created that would deal adequately with the structure (and information content) of all the layers of detail. This decomposition process can be applied to each column of the framework, where the mechanisms for instantiation of each cell are contained and are part of the FDA.

An *engine* is defined as an analysis object that represents and implements a complex business capability requiring the integration of knowledge, decision models, and data resources of various types. It can

be thought of as a large block of reusable application code that instantiates complex business functions. Engines are shareable and reusable in disparate business contexts.

To conduct business using the AID and AIO, we propose a Virtual Business Environment (VBE). A business environment is considered *virtual* when only its spacial or temporal instantiation (and not its description or model) depends upon the physical surroundings and locations of its actors, decision contexts, capabilities of the devices being used, and the actors' entitlements. VBE is an umbrella term for a collection of technologies that allow an enterprise architecture to exist as one seamless unit. A VBE requires the management of storage devices, servers, and network devices through a central console. Behind the console, high-end database management systems from Oracle, for example, should coexist with open-source database managements systems such as MySQL**. Similarly, high-end Sun servers should coexist with low-cost blade servers. In this way, multiple and independent environments can use a single set of resources. VBEs allow the architecture to become responsive (i.e., liquid) to unknown requirements.

Engines are used by the VBE, which requires their functionality. The engine needs to integrate resources, synchronize models and data, normalize outputs, dynamically allocate resources, and enforce business rules. Engines may be added to or expunged from a VBE. An engine manager (another instance of an engine) maintains a library of engines and identifies appropriate engines and the resources necessary to run them.

An enterprise may have many such environments (in the process domain of its FDA), and their boundaries may be established on an instance-by-instance basis. The business processes (or subprocesses) are defined in a VBE based upon the needs of the decision makers and users. The VBE would allow organizations to run different business processes concurrently and securely, while sharing resources. For example, a VBE can support concurrent availability of different business views (e.g., one for senior executives and another for project managers, both sharing data, models, and business contexts).

Conceptually, the VBE consists of a domain-resource subsystem, a subsystem of engines, and a dialog-management subsystem. The domain-resource subsystem manages structured and subtly structured (nontabular) data, including expert knowledge, models, documents, and data needed for decision making. The dialog management subsystem is responsible for interacting with the user to query information, provide inputs, and interpret the responses from the engines to create individualized outputs for the user. The outputs can be rendered in different ways as specified in the user profile.

The process domain of the FDA also typically contains software components called "business context engines." Examples of these include business-management environments, business-process-development environments, and data-control environments.

Table 2 shows the elements we would expect to find if we were to document a process (the "how" column of the framework) for each of the four domains of the FDA.

We will now apply the decomposition process to two columns (data and function) of the Zachman framework.

Framework cell: Column: Function, Row: Scope

This cell could be occupied by a list of action verbs that would categorize all of the business processes that would be needed by the enterprise being documented in this framework instance. These action verbs, supported by appropriate meta-data, would be part of the information/knowledge domain artifacts related to this particular cell.

There would be a specific process (known as a modeling process in the terminology of enterprise functional decomposition[3]) that would produce this list of action verbs and their meta-data as its output. Using this process, a planner can develop an understanding of the overall functions, activities, and components of the enterprise. For example, the planner could use techniques such as value chain analysis,[24] enterprise functional decomposition, or business web analysis[25,26] for this purpose. This process would be a component of the architecture's process domain. It might be collocated with other processes in a strategic planning environment (an instance of a VBE), and may make use of the facilities of several engines, such as an inference engine, an induction engine, a library engine, and a modeling engine. This process itself would be created by a process for building business processes, which would be, in turn, part of the business process development environment (BPDE, another instance of a VBE) of the architecture. The subsetting process also would be a process of a VBE

Table 2 Expected elements in process documentation

Process Domain Contains:	Information/Knowledge Domain Contains:	Infrastructure Domain Contains:	Organization Domain Contains:
The process being documented itself, and all other processes that are involved in the effort at this level of detail	The names and functional descriptions of the processes	The subsets of the infrastructure domain that are the essential platforms for supporting its processes	The subsets of the organization domain that are the essential actors (owner, user, partner, ally) for the processes
The Virtual Business Environments (VBEs) that "contain" the processes and the engines that participate in the instantiation of the functionality of these processes	The business rules that the processes embody	At the appropriate level of abstraction, the features and functionality (e.g., security requirements, reliability, availability, responsiveness) that are needed to effectively instantiate the processes	At the appropriate level of abstraction, the roles and responsibilities that are needed to effectively govern the processes
	The business terms that relate to the processes	The characterization of the infrastructure-domain sensory networks related to the processes	The "spheres of influence" that characterize the business webs related to the processes
	The meta-data related to the processes and the data subject areas that drive or are produced by the processes at the appropriate level of abstraction		

(strategic planning or portfolio management), which would produce priorities among the processes identified.

The data resources required to support this framework cell would be part of the information/knowledge domain of the architecture. Data requirements would be gathered and mapped to or integrated with the data requirements of the other components of the various domains. These requirements would establish the information/knowledge context of the enterprise by describing the processes of the VBEs of the process domain (e.g., the data control environment) that manages the information/knowledge domain of the architecture. These data requirements would be cataloged and stored as part of an appropriate information structure in this domain.

The platform resources required to support this cell would be part of the infrastructure domain of the architecture. Technical requirements and data needs would be gathered and mapped to or integrated with the technical requirements and data needs of the other components of the various domains. These requirements would establish the technical context for the emerging "virtual information factory" of the enterprise by describing the processes of the VBEs of the process domain (e.g., the information-services-operations environment and the security environment) which manage the infrastructure domain. These technical requirements would be cataloged and stored as part of an appropriate information structure in the information/knowledge domain.

The roles, responsibilities, and applicable business rules for individual actors and operating units *vis-à-vis* this framework cell, its supporting processes, and their inputs and outputs would be articulated within the organization domain of the architecture. Included would be the definitions of the roles and responsibilities for the framework's row owner (i.e., planner), for the other participants in the definitional work for this cell, and for the owner of the framework itself. These actor and unit attributes would be developed and maintained by using appropriate process domain resources (i.e., environments, processes, and engines). The organizational context of the enterprise would also be established by describing the processes of the VBEs of the process domain (e.g., the enterprise control environment) that man-

age the organization domain of the architecture. These requirements would be cataloged and stored as part of an appropriate information structure in the information/knowledge domain.

Framework cell: Column: Function, Row: Enterprise model

This cell could be populated by a set of functional flow diagrams, which would document a selected subset of the functions, activities, and components articulated in the cell above. These flow diagrams, supported by appropriate meta-data, would be the information/knowledge domain artifacts that satisfy the purposes of this particular cell.

There would also be additional specific processes that would produce these flow diagrams and their meta-data and document their inputs and outputs. The owner could use functional decomposition diagrams to understand the selected subset of the enterprise at this level of detail. This process would also be a component of the architecture's process domain. It might be collocated with other processes in a BPDE and might make use of the facilities of several engines. This process itself would be created by a process for building business processes, which also would be part of the BPDE of the architecture.

Framework cell: Column: Function, Row: System model

This cell could be populated by a set of flow diagrams, which would document the business systems (processes) identified earlier. These flow diagrams (e.g., data-flow diagrams), supported by appropriate meta-data, would be the information/knowledge domain artifacts that fulfill the needs of this particular cell. A specific set of design processes would be used to produce these flow diagrams and their meta-data. The designer could use tools such as data-flow diagrams, state-transition diagrams, and so forth.

Framework cell: Column: Function, Row: Technology model

This cell may contain a set of process structure charts that describes how the process is to be constructed, the critical functionality of each building block, and the related inputs, outputs, and controls. For this row, the information/knowledge domain artifacts may be process charts or pseudo-code, supported by appropriate meta-data. The builder could use tools such as HIPO (Hierarchical Input Process Output) charts, flow charts, decision tables, and so forth, as design processes to produce the process structure charts.

Framework cell: Column: Function, Row: Components

A specific set of authoring/generating processes and procedures would be used to produce these artifacts and their meta-data, which include programming environments and pseudo-code. In addition, other processes, provided by other environments, might be required to test the deliverables of this cell, provide configuration management and integration services, and so forth.

Framework cell: Column: Data, Row: Scope

According to the framework, this cell could contain a list of all the things (i.e., material) that are important to the business and managed by it; for example, entity classes, products, parts, supplies, equipment, employees, customers, suppliers, competitors, buildings and real estate, policies and procedures, purchase orders, and so forth. The meta-data about these entities of interest is part of the information/knowledge domain artifacts related to this particular cell.

The architecture should contain a process that would help select the entity class or classes in which to invest actual information-systems resources for the purposes of data inventory management. This process would help clarify the values and strategies of the business, using various data gathering and analytical techniques. After applying the analysis to the total set of entity classes, a subset of classes may be selected for implementation.

The infrastructure domain of the architecture includes the functionality, such as security, availability, and responsiveness, that is necessary to effectively instantiate the processes just described. This domain includes the sensory networks necessary to collect data from the environment to support the selection of the entity classes. The organization domain for this cell includes the roles, responsibilities, and applicable business rules for individual actors and operating units, such as the team that decides which entity class is selected for implementation.

Framework cell: Column: Data, Row: Enterprise model

This cell contains a description of the business entities and their relationships. The relationship in this case would be the business rule or strategy that links

one entity to another. The owner describes the business entities and their relationships by using business rules. For example, in a particular business setting, a premium customer is always assigned to a unique relationship manager. Another example is a stipulation by the owner that all the different businesses must be first or second in their particular market space. The architecture should contain a process that would help describe the entity class or classes in which the owner has an interest. This process would help clarify the importance of key entities and the business rules that link them. After analysis is applied on the total set of entity classes, a subset of classes is selected for implementation.

The infrastructure domain of the architecture includes the functionality, such as security, availability, and responsiveness, that is necessary to effectively instantiate the processes.

Framework cell: Column: Data, Row: System model

This cell contains the model of the information system in the descriptive form of entities and relationships. The designer may think of an entity as an information item on a computer data store and a relationship as a data relationship. For example, every identified market space has a well-defined set of competitors, growth rates, and total revenue forecasts. The architecture should contain a process that would help the designer describe these entities and their relationships. The data resources required to support this cell would include meta-data about the entities and their relationships.

Framework cell: Column: Data, Row: Technology model

This cell would contain the description model and the data design for the conceptual model of the information system. In this model, various constraints that are to be applied are specified. For example, the builder could specify that every market space has three forecasts for growth rates (optimistic, average, and pessimistic). The decision about which alternative database management system to be used is also specified here.

Application of AIO and AID using the Zachman framework

In the previous section, we discussed how the FDA can be used to instantiate the various cells of the Zachman framework, and we illustrated the recursive nature of this exercise within columns of the framework. Here, we illustrate how the subarchitectures (the AID and AIO) of the FDA are to be applied to the framework, and we show that the outcomes are consistent with the results achieved by applying the entire FDA to the same cells. Similar concepts have been discussed in Reference 27, which, in addition to providing an excellent introduction to the Zachman framework, introduces the concept of primitive and composite models that can be used to implement the subarchitectures (AID and AIO) described here.

In the FDA, each cell of the Zachman framework is split into two—the AID and AIO subcells. In the following subsection, we present as an example of the FDA several subcells used in the definition and creation of a process (and related tool set) for the design of an electronic product to be manufactured by the enterprise's manufacturing organization.

AID subcells. We describe here two AID subcells of cells described in the previous section.

Framework cell: Column: Function, Row: System model

This subcell could be populated by a set of flow diagrams, which would document the product design system.[28] The design processes used to create flow diagrams to populate this AID subcell would be components of the process domain of the architecture, collocated with other processes in a BPDE. These design processes themselves would be created by a process for building business processes and their supporting tool sets, which are also part of the architecture's BPDE for this subcell.

The infrastructure required to support this cell would be part of the infrastructure domain of the AID. Technical requirements and needs would be gathered and mapped to or integrated with the requirements and needs of various other AID-related domain components and implemented by the infrastructure domain services of the AID.

In other words, the AID components produced for this cell of the framework would contain the models for the emerging business processes needed to design an electronic product. These models would be stored in the AID information/knowledge domain along with the product-design process meta-data. All of this activity would be carried out or supported by specialized subsets of the infrastructure and organization domains of the AID that are involved in the work of designing the processes for designing products.

Framework cell: Column: Data, Row: System model

This subcell contains the model of the information system in the descriptive form of entities and relationships. For example, every electronic component has a well-defined set of physical, electrical, and operational characteristics, and these characteristics would give rise to entity and relationship definitions. The process domain of the FDA (for the AID) should contain processes that would help the designer describe the entities and their relationships. Other tools in the AID might test candidate entities and relationships for coherence with existing entities, their definitions, process models and other data models, and so forth.

AIO subcells. We describe here two AIO subcells of cells described in the previous section.

Framework cell: Column: Function, Row: System model

This AIO subcell could be populated by a set of electronic-product design artifacts (circuit schematics, design diagrams, performance charts, etc.) which would document the emerging product design. The information/knowledge domain artifacts for this subcell are produced in the electronic product design environment of the AIO. This environment is instantiated from (1) the models of the product design process produced by the tools and (2) the process for designing product design processes of the BPDE for the AID.

The electronic product designer (e.g., an electronic design engineer) would use the instantiated electronic design tools (e.g., computer-aided-design tools) and design processes for producing the product design. These design tools and processes would also be components of the AIO process domain of the FDA, collocated with other processes in a product engineering environment.

In other words, the AIO instantiation components produced for this cell of the framework would contain the business processes needed to design an electronic product. These instantiated processes would be stored in the information/knowledge domain of the operational module along with the meta-data and data of the design processes. All of these activities would be carried out or supported by specialized subsets of the infrastructure and organization domains of the AIO that are dedicated to doing the work of designing products.

Framework cell: Column: Data, Row: System model

This cell contains the instantiated data model (in the AIO) required to operate and support the product design process of the AIO. The electronic product designer thinks of these data as design variables or parameters that are fundamental inputs or aspects of the product design, for example, the set of physical, electrical, and operational specifications for every circuit to be considered for possible inclusion in the emerging design. The electronic-product-design environment of the process domain of the AIO should contain a process to help the designer create specific electronic libraries and populate them. Other tools in the AIO might generate component taxonomies or search criteria, or allow for the generation of simulator programs to be used in the testing of the emerging design.

Conclusion

The artifacts chosen to capture and represent activities within each one of the cells within the Zachman framework are left to the discretion of the enterprise, making it difficult to impose rigor or share knowledge across enterprises. The FDA model proposed in this paper helps in the design and building of a tool set and methods to support frameworks such as Zachman's. Rigor is added to the process by considering both the AID and AIO, and tracking the difference between them over time.

A key benefit of this approach is the ability to assess how a project may impact the emergent architecture via a process. Observing the emergent architecture and evaluating it in light of the project requirements will enable an enterprise to fine tune the impact of the project on the architecture, thus improving the design and facilitating decision making concerning which projects to implement in a constrained environment. In addition, the FDA approach will help enterprises synchronize their business needs and IT

capability through conscious changes to the emergent architecture, thereby improving project performance in both the near term and the long term.

A separate issue not addressed in this paper is the activity of harvesting architectural assets gleaned from different projects. This knowledge management imperative requires consistency of terminology and notation to create and utilize engagement artifacts. For an overview of this issue, see Reference 29.

Researchers have posited that architectures have three value-creating potentials: responsiveness, innovativeness, and economies of scope.[17] Companies with a better-designed architecture will be more responsive in adapting their knowledge repositories to accommodate changing business conditions than those with a poorly designed architecture. A flexible architecture provides a foundation for innovation in light of uncertain future knowledge requirements. In addition, such an architecture allows experimentation with innovative information technologies.

The third value-creating potential, economies of scope, refers to the ability of architectures to reduce IT-related costs for the development of knowledge repositories. As a result, the development cost and time to develop knowledge management systems for repositories should be lower for firms with more capable architectures than for firms with less capable architectures.

We believe that creating an environment to support the FDA model will enable organizations to capture the anticipated benefits from architecture design, development, and use.

**Trademark or registered trademark of MySQL AB Company.

Cited references and note

1. E. Yourdon and L. Constantine, *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*, Yourdon Press, Englewood Cliffs, New Jersey (1986).
2. C. Gane and T. Sarson, *Structured Systems Analysis: Tools and Techniques*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1979).
3. T. DeMarco, *Structured Analysis and System Specification*, Yourdon Press, Englewood Cliffs, New Jersey (1979).
4. R. Sanchez, "Modular Architectures in the Marketing Process," *Journal of Marketing* **63,** No. 5, 92–111 (1999).
5. C. Y. Baldwin and K. B. Clark, *Design Rules: Volume 1. The Power of Modularity*, MIT Press, Cambridge, MA (2000).
6. M. W. Maier and E. Rechtin, *The Art of Systems Architecting*, CRC Press, Boca Raton, FL (2000).
7. C. Alexander, *Notes on the Synthesis of Form*, Harvard University Press, Cambridge, MA (1964).
8. M. R. Vigder and J. C. Dean, "An Architectural Approach to Building Systems from COTS Components," *Proceedings of the 22nd Annual Software Engineering Workshop*, Goddard Space Flight Center, National Aeronautics and Space Administration (1997).
9. J. A. Zachman, "A Framework for Information Systems Architecture," *IBM Systems Journal* **26,** No. 3, 276–295 (1987).
10. K. T. Ulrich and S. D. Eppinger, *Product Design and Development*, Second Edition, McGraw-Hill Companies, Inc., Boston, MA (2000).
11. M. Meyer and A. Lehnerd, *The Power of Product Platforms: Building Value and Cost Leadership*, The Free Press, New York (1997).
12. We wish to thank John Zachman, who, in a private communication about our architecture on September 28, 2002, suggested this approach.
13. J. Ross, "Creating a Strategic IT Architecture Competency: Learning in Stages," *MIS Quarterly Executive* **2,** No. 1, 31–43 (2003).
14. D. McKay and D. Brockway, "Buidling IT Infrastructure for the 1990s," *Stage by Stage* **9,** No. 3, 1–11 (1989).
15. N. Duncan, "Capturing Flexibility of Information Technology Infrastructure: A Study of Resource Characteristics and Their Measure," *Journal of Management Information Systems* **12,** No. 2, 37–57 (1995).
16. P. Weill and M. Broadbent, *Leveraging the New Infrastructure: How Market Leaders Capitalize on Information Technology*, Harvard Business School Press, Boston, MA (1998).
17. T. Kayworth, D. Chatterjee, and V. Sambamurthy, "Theoretical Justification for IT Infrastructure Investments," *Information Resources Management Journal* **14,** No. 3, 5–14 (2001).
18. R. Gottlieb and R. McCluskey, *Factory Mutual: Information Technology Architecture*, Factory Mutual: Boston, MA (2000).
19. R. Cyert and J. March, *A Behavioral Theory of the Firm*, Prentice-Hall, Englewood Cliffs, NJ (1963).
20. R. Nelson and S. Winter, *An Evolutionary Theory of Economic Change*, Harvard University Press, Cambridge MA (1982).
21. R. M. Henderson and K. B. Clark, "Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms," *Administrative Science Quarterly* **35,** No. 9, 9–30 (1990).
22. J. A. Zachman, "The Framework for Enterprise Architecture (The "Zachman Framework") and the Search for the Owner's View of Business Rules," *Database Newsletter* **1,** No. 26 (1998).
23. J. F. Sowa and J. A. Zachman, "Extending and Formalizing the Framework for Information Systems Architecture," *IBM Systems Journal* **31,** No. 3, 590–616 (1992). See also http://www.zifa.com/framework.html.
24. M. Porter, *Competitive Advantage*, Free Press, New York (1985).
25. D. Bovet and J. Martha, *Value Nets: Breaking the Supply Chain to Unlock Hidden Profits*, John Wiley & Sons, Inc., New York (2000).
26. S. Cartwright and R. Oliver, "Untangling the Value Web," *Journal of Business Strategy* **21,** Vol. 1, 22–27 (2000).
27. C. O'Rourke, N. Fishman, and W. Selkow, *Enterprise Architecture: Using the Zachman Framework*, Course Technology, Boston, MA (2003).
28. P. Balasubramanian, K. Nochur, J. C. Henderson, and M. M. Kwan, "Managing Process Knowledge for Decision Support," *Decision Support Systems* **27,** Nos. 1–2, 145–162 (1999).

29. R. Youngs, D. Redmond-Pyle, P. Spaas, and E. Kahan, "A Standard for Architecture Description," *IBM Systems Journal* **38,** No. 1, 32–50 (1999).

Accepted for publication February 26, 2004.

Bala Iyer *Assistant Professor, Boston University School of Management, 595 Commonwealth Avenue #641A, Boston, Massachusetts 02215 (bala@bu.edu)* Professor Iyer is an Assistant Professor of Management Information Systems in the Department of Information Systems, Boston University. He received his Ph.D. degree from New York University with a minor in computer science. His research interests include designing knowledge management systems by using concepts from systems design, hypertext design, and workflow management, by exploring the role of IT architectures in delivering business capabilities, and by querying complex dynamic systems, hypermedia design and development, and model management systems. Recently, he has begun to analyze data on the software industry to understand the logic and patterns of emergence of software architecture from multiple perspectives. He has published widely on information systems and operations research. In 2003 he won the IBM faculty award.

Richard M. Gottlieb *Former Executive in Residence, Boston University School of Management, 60 Ruggles Street, Westborough, Massachusetts, 01581 (richgott@localnet.com).* Mr. Gottlieb received his M.S. degree from Rutgers University in 1961 and an S.B.E.E. degree from the Massachusetts Institute of Technology in 1958. During a career of more than 40 years in the United States and abroad, he has been actively engaged with almost all aspects of computer and information technology, from systems design, marketing and systems sales, to the management of information services. He has had hands-on experience with reengineering corporate IT functions and with the introduction of high-level architectures and advanced technologies into the enterprise. Before joining the faculty of Boston University, he was the Vice President of Information Services at Factory Mutual Engineering and Research, where he articulated, promoted, and implemented many of the Four-Domain-Architecture concepts upon which this paper is based. Mr. Gottlieb is now retired.