Foreword to Third Edition

I was honored to write the Forewords to both the First and Second Editions of Clive Finkelstein's book, *"Enterprise Architecture for Integration: Rapid Delivery Methods and Technologies."* I liked the Forewords I wrote for the earlier versions so much that I kept them unscathed below! Clive explains that his Third Edition principally adds an Appendix that is based on a paper he wrote entitled *"What Is Wrong with Information Engineering and How to Fix it."*

First of all, let me observe that I was there during the history that Clive very graciously describes when the misunderstandings about Information Engineering were taking place and it was not Clive that had anything to do with *"what was wrong with Information Engineering!"* In fact, Clive had everything to do with what was RIGHT about Information Engineering. I will try to be as gracious as Clive has been in describing my perceptions.

One principal problem was cultural in nature. The world was simply not ready for Clive's concepts and it may well still have some of those same cultural tendencies. It seems that many people tend to be process dominant. We can even see some of these cultural characteristics linguistically in the romance languages in being verb oriented, a rich capability in articulating process. In contrast, English, for example is noun oriented, a rich capability for describing things, that is, in the technology vernacular, *"entities"*, or data. So, in *"Data Processing",* there are two kinds of people: data people and process people.

In the late '60's, NASA had put people on the moon and *"Systems Engineering"* was attributed much of the success. In fact, as I remember, the *"Systems Engineer"* was on the cover of Time Magazine, the Time *"Man of the Year".* The orientation was the systematic process of doing the engineering to make the moon landing a reality.

This next observation may be folklore as I am not able to validate it, but my understanding is that John von Neumann actually invented two different kinds of computers, the process computer and the data computer. (I did not know John von Neumann but I have a friend that did know him and I will ask my friend if he can validate this observation for me.) In any case, in a Process Computer, you put the process in the computer and pass the data past it. In a Data Computer, you put the data in the computer and pass the processes past it. At the time, the Process computer became commercially successful as the scientific community was processing nuclear code, complex processing algorithms and minimal data, and the business community was automating processes, that is, replacing people with machines for doing the (systems) processing and the data simply was what was needed to get the code (the system) to run.

Clive made a radical observation … if you define the data relative to the processes of the enterprise, you will de-normalize, that is, DIS-integrate the data and lose control of the asset structures and furthermore, the resultant systems will have a short life, only as long as the life of the systems processes, which are relatively dynamic. On the other hand, if you derive the data from the asset structures (the *"entities"*) you can normalize (that is, *"integrate"*) the data and the resultant systems will have a relatively long life, as long as the asset structure life, which is relatively stable and you will support enterprise Management's mission and objectives as well.

The culture, and especially the technology culture (Data Processing) was not ready for Clive's radical observations … and even today, the popular focus is still on getting the code to run, typically at the expense of normalizing (integrating) and managing the data … at the enterprise expense of losing control of the business assets and not meeting management's expectations.

In fact, the state of the art Data Processing methodologies in the 70's and 80's were called *"Systems Engineering"* after the NASA successes. I am sure Clive knew that he was not talking about *"more of the same",* that is, he was not talking about *"Systems Engineering."* Therefore, he named his methodology *"Information Engineering"* in order to differentiate it from the current state of the art at the time.

There was a second, complicating factor contributing to the mis-understanding of Information Engineering. In the original manuscript of the 1981 *"Information Engineering"* book co-authored with James Martin, Clive described the data orientation of his methodology and how the enterprise Business Processes could be derived from the resolution of the *"many-to-many"* Entity relationships, the *"intersection"* entities. By definition, there HAS to be a Business Process that manages every intersection Entity. *Unfortunately, by the time the first edition of the 1981 book was actually published, all of Clive's innovative material had been removed from the book and replaced with the popular process-orientation … for marketing reasons, I am sure.*

So, it is not Clive's fault that Information Engineering was understood to have a process-orientation at its very outset rather than the intended data-orientation!

Clive spent many years trying to rectify this misunderstanding. I hope this *"What Is Wrong with Information Engineering"* article will get people's attention and I hope for this iteration the world is ready to hear the facts and embrace the real, original, Information Engineering logic!

With this somewhat lengthy preamble, let me proceed with my original Forewords …

Clive Finkelstein has authored the second edition of his landmark book, *"Enterprise Architecture for Integration: Rapid Delivery Methods and Technologies."* Clive has used my Framework (the *"Zachman Framework")* to contextualize his methodology, which I appreciate immensely. One of Clive's motivations for writing the second edition was to update references to my Framework to reflect the present Framework vocabulary.

When I first articulated the Framework graphic, I used words for the models and metamodels that came from my vocabulary, an information-based vocabulary. Unfortunately, by so doing, I contributed to the misperceptions that Enterprise architecture is an Information Technology issue as opposed to an ENTERPRISE issue.

Let me be clear … Enterprise Architecture IS an ENTERPRISE issue, not an Information Technology issue. *This is fundamental to Clive's methodology.* Surely, Information Technology and the resultant systems implementations are not going to go away because they are one possibility for implementing the Enterprise. (Manual systems or no systems at all are also possibilities for implementing the Enterprise … each of which has its place and its significant implications.)

Over the years, I have modified the vocabulary of the Framework Graphic to better reflect the classification concepts it represents and also, more importantly, to neutralize the Information Technology connotations. I have tried to reflect the classification concepts clearly and precisely and to do it in such a way as to represent an Enterprise, independent of its implementation choices.

The Framework is the Framework is the Framework. The classification on each axis of the Framework has been employed by humanity for literally thousands of years. These classifications have not changed and they are not going to change. My vocabulary has improved and my word choices are more neutral and I appreciate Clive's efforts to reflect the present vocabulary, which was a non-trivial piece of work on his part!

Clive Finkelstein and I have been good friends for over 30 years. I have a great respect for his wisdom and thoughtfulness. My opinion is, he has made profound contributions to the Information Profession and this book is another contribution of such significance.

I wrote the Foreword to the first edition of the book and I actually like that Foreword. I have not changed my mind appreciably since I wrote it and therefore, I am going to recreate that Foreword with a few editorial improvements and some additions and deletions as follows …

I cannot sufficiently impress you with the significance of Clive Finkelstein's book *"Enterprise Architecture for Integration: Rapid Delivery Methods and Technologies",* First Edition. Those of us who are alive and in the workforce in 2015 are the transition generation, in transit from the Industrial Age to the Information Age, much like those of several centuries ago as the world changed from the Agricultural Age to the Industrial Age. Hopefully, many of us will be survivors in the massive, global revolution that is currently being waged: the Information Revolution. I always have said that the best place to be in a revolution is on the winning side. *Clive Finkelstein is providing a strategy that will position an enterprise to be on the winning side.*

The academics make the case that it takes a lifetime to make the transition from one global environment to another. The scientific measurement for one lifetime is a 40-year life cycle and there is some evidence that we began the transition somewhere between 30 and 35 years ago. That would suggest that we might have only 10 or 15 years to go before we will know who has successfully made the transition and who has been left behind. There is a case to be made that the revolution is much longer than 40 years. It is a classic four-stage learning curve but a learning curve of the whole of humanity. For example, when did the Industrial Revolution start? How long was it? It probably started between 3 and 4 hundred years ago and it continued for around a hundred years with vestiges of it still occurring. Nonetheless, it is likely that the players and the non-players will be evident in the not too distant future.

Those of us who are alive in 2015 are the transition generation. We are the last of the Industrial Age people. Although none of us knows positively all of the Information Age characteristics, much has been speculated and written about. Likely, the most widely read and credible series of books were those authored by Alvin Toffler (*Future Shock,* Random House, 1970; *The Third Wave,* William Morrow, 1980; and *Powershift*, Bantam Books, 1990). Thanks to him and others, not to mention our own personal experience, our understanding of some of the characteristics of the Information Age is taking shape.

First, major changes clearly are taking place. Business is no longer going to be as simple as *"get yourself a good product or service and then go find a bunch of people to sell it to."* The Information Age business is *"get yourself a good customer and then find the range of products and services required to keep that customer a good customer."* That is a lot more complicated! The Information Age business is far more complex than Industrial Age businesses. The day you have to treat each customer as an individual and customize (integrate) the enterprise response to the customer requirement, you are signing up for orders-of-magnitude increases in complexity. From the perspective of the enterprise, it is no longer a case of the market being integrated. Now it is, from the perspective of the customer, the enterprise must be integrated.

Those who have to deal with integration, customer or enterprise, have to accommodate the complexity. The concept of *"stovepipes"* is anathema to the Information Age enterprise. The enterprise is going to have to be integrated. How do you suppose an enterprise is going to get integrated? By accident? By writing some more code? By wishful thinking? I submit, the people who are building the enterprise systems ... automated and manual ... are going to have to produce enterprise-wide, integrated implementations if there is ever going to be any enterprise-wide integration. Complexity is escalating dramatically. No longer is it adequate just to get the manual and/or automated systems to work.

A second characteristic of the Information Age that is becoming abundantly evident is the dramatic escalation of the rate of change. We are all running out of time. I am running out of time. You are probably running out of time. IT is running out of time. The enterprise is running out of time. From a consumer (customer) perspective, we all need *"custom products, mass-produced in quantities of one for immediate delivery"* because we do not know the nature of the product we want to take delivery on until we want to take delivery on it.

From the supplier (enterprise) perspective, you cannot wait until you get the order to engineer and manufacture your response. You cannot anticipate and manufacture and have in storage every finished good the consumer is ever going to want to take delivery on. You are going to have to engineer parts (not finished goods), prefabricate them and have them in inventory before you ever receive an order … and the parts are going to have to be cleverly engineered such that they can be assembled into more than one finished good. The engineering will have to be done *"enterprise-wide."* The manufacturing name for this concept is *"mass customization"*.

The same conceptual approach will be required for the enterprise as for any product if enterprise management are unable to define the nature of the enterprise implementation they need until the moment they need it. Those of us who are engineering and manufacturing the enterprise (that is, building and running systems) will have to respond with a custom enterprise, mass-produced in quantities of one for immediate delivery (mass-customization of the enterprise). Also, regardless of whichever implementation management wants, automated or manual, they will want it integrated, enterprise-wide. This will be particularly critical for service-based enterprises because the enterprise services are simply the enterprise as viewed from the perspective of the customer.

In any case, some principal characteristics of the Information Age we know of so far are extreme complexity and extreme rates of change. The question for the enterprise is, *“How do you intend to deal with orders-of-magnitude increases in complexity and orders-of-magnitude increases in the rate of change?”* Do you think this is not happening? The question is not, *“Is this going to happen?”* The only question is, *“What are you going to do about it?”*

Seven thousand years of known history of humankind would suggest that the only known strategy for accommodating complexity and change is architecture. If it (whatever it is) gets so complex that you can’t remember all of the details all at one time, you have to write it down ... describe it ... architecture. If you cannot describe it, you cannot create it. After you get it (whatever it is) created and you want to change it, how do you change it? You start with what you wrote down ... architecture. The key to complexity and change is architecture.

I submit that if you do not have an enterprise architecture strategy, you do not have a strategy for addressing orders-of-magnitude increases in complexity and orders-of-magnitude increases in the rate of change and therefore, you are likely to have a very difficult time maintaining your viability in an Information Age environment.

Clive’s book is not about the Zachman framework. In fact, Clive refers to other books about the Zachman framework. Clive’s book is a methodology book. He provides a summary overview of the framework graphic because he maps his methodology against the framework. The framework provides a context within which the logic of a methodology can be expressed in an understandable manner. Furthermore, the framework provides a structured definition of enterprise architecture that Clive exploits, whereas historically enterprise architecture is an issue that most methodologies have tended to ignore.

I learned about the Framework by looking at architectural representations of tangible objects like buildings, airplanes, locomotives, computers, etc., etc. It is all the same. Because I was interested in engineering and manufacturing enterprises, I simply put enterprise names on the same descriptive representations found in the older disciplines of Architecture and Construction and of Engineering and Manufacturing.

In the Industrial Age, the enterprise value proposition for computers was *“better, faster, cheaper.”* Computers were better than people because people make mistakes and computers do it the same way every time. Machines are faster than people and machines are cheaper than labor in most cases. If *“better, faster, cheaper”* is the enterprise value proposition, it will drive you to a very short-term approach for implementation because the very moment you discover something repetitive going on in the enterprise that is not automated, it is actually costing you quality, time, and money (better, faster, cheaper). There is an incredible incentive to get the systems implemented as soon as possible. Anything that inhibits implementation is *“analysis/paralysis.”* This value proposition has spawned a whole genre of rapid application development-style methods and tools.

In contrast, if accommodating extreme complexity and high rates of change are the enterprise value proposition, then the engineering design objectives are integration, flexibility, alignment, interoperability, reduced time-to-market, security, and so forth…not simply implementation. The Information Age value lies in creating an inventory of knowledge about the enterprise, knowledge assets, enterprise architecture knowledge assets, "primitive", single-variable models, single cell models in Framework terms. These primitive models must have been engineered for integration, flexibility, alignment, and so forth before there is a requirement for any implementation. From such an inventory of primitive models, virtually any manifestation of the enterprise could be assembled to order by the click of a mouse. Now we would be talking REALLY rapid application development ... however, somebody has to get the primitive models engineered and into inventory before an order for such is received.

This is all physics. Nothing happens by accident. If you want integration and reusability, then you are going to have to build enterprise-wide primitive models. If you want alignment, then you are going to have to define what you are aligning to, that is, the higher row models. If you want flexibility, then you are going to have to build primitive models and keep them separated until you want to implement them. If you want to accommodate change, then you are going to have to build primitive models and retain them to serve as a baseline for managing change. If you want to reduce the time it takes to implement systems, then you are going to have the primitive models in inventory before you get an order for implementation. If you want quality, then you are going to have to make the primitive models explicit and make them explicit at excruciating levels of detail. And so on. Let me give you some friendly advice: There is no such thing as a free lunch ... and there are no "*silver bullets" ("No Silver Bullet"* by Fred Brooks 1986 reprinted in *"The Mythical Man Month"* Addison Wesley 1995).

On the other hand, if all you want is implementation as soon as possible, then *"you start writing the code and I'll go find out what the users have in mind"* (the caption on an old cartoon) later … which will be more of the same that we presently have, a *"legacy"*.

Although I have known Clive Finkelstein for more than 30 years, and although we were coming from entirely different perspectives … he from a methodology perspective and I from an architecture perspective … we were both arriving at the same conclusions. The end object is not simply to get the systems to run and, therefore, it is not adequate simply to write code. Engineering work, ENTERPRISE engineering work has to be done because the enterprise problem in the Information Age is integration, flexibility, reusability, quality, security, reduced time to market, and so forth.

Yes, you have to get to implementation as soon as possible, but if you are not assembling the implementations from the primitive models that are already in inventory, that is, enterprise architecture, all you are going to end up with is more code, that is, more legacy. And, it makes no difference what technologies you are using, how current your operating system is or how clever your programming, you are just building more legacy that is not integrated, not flexible, not reusable, not secure, not aligned, not quality, and not very rapid either!

Clive's methodology is one of the few methodologies that I know of in 2015 that actually addresses some of the enterprise engineering design objectives that go far beyond just getting the code to run. I hope you can see why I said at the outset that I could not sufficiently impress you with the significance of this book.

A lot of work needs to be done and we are running out of time. At the point in time at which the enterprise is critically dependent on accommodating extreme complexity and extreme rates of change, it is going to be too late to start working on enterprise architecture because enterprise architecture is not simply one more implementation project. Enterprise architecture is a different way of life. At least some enterprise architecture work is going to have to be in place before the enterprise urgently requires it. I recently asked someone in one of my seminars how long they thought they had before they had to have some major pieces of this in place ... 20 years? ... 10 years? ... 5 years? Their response was *"We needed this 2 years ago!"* It will not be long before we will know which enterprises have gotten some of this enterprise architecture work done ... and which haven't!

I hope you enjoy Clive's book as much as I have.

John A. Zachman
Chief Executive Officer
Zachman International
Glendale, California
February 2015