

You Can't "Cost-Justify" Architecture

By
John A. Zachman
© Copyright 2001 Zachman International

Introduction

Wherever I go and talk about Enterprise Architecture, the most frequently asked question I get is, "Well, how do you 'cost-justify' Architecture?" The reason I keep getting this question is, people are having a difficult time "cost-justifying" Architecture. The common perception is, it takes too long and it costs too much to do Architecture and besides that, you have to do too much work before you can deliver some end results. Anyone who has these perceptions simply does not understand the concept of, nor the implementation strategies for, Architecture, nor do they understand that the "Industrial Age" is now and forever over and that *the game has forever changed!*

"Cost-justification" is an idea that came out of the Industrial Age when the value proposition for computers ("systems") was "better, faster, cheaper." Computers were better than people because computers do things the same way every time and people make mistakes. Computers are faster than people because computers operate at electronic speeds and people operate at people speeds. And, computers are cheaper than people because machines are cheaper than labor. "Better, faster, cheaper." It is "better, faster and cheaper" to displace people costs with computer costs. Improve quality, reduce time and save money. Increase per-unit productivity. You could justify the acquisition expense of a computer based on displacing other (labor) costs. "Cost-justification," an *expense* based concept. And, it worked. The per-unit productivity, particularly in the United States, rose to unprecedented heights. In fact, there is no way an Enterprise of today can compete in the marketplace without having their basic functionality automated ("computerized").

But ... the Industrial Age is over. I recently read that investments in computer technology continue to escalate dramatically but unexplainably, productivity statistics are virtually flat!¹ I would suggest that basically, we have displaced all of the obvious labor

¹ Loveman, G. (1991). "Cash Drain, No Gain," Computerworld, 25(47) 62-69. And, Roach, S.S. (1991) "Services Under Siege – The Restructuring Imperative," Harvard Business Review 69(5) 82-91.

costs with computer costs and therefore “cost-displacement” (that is, “cost-justification”) is no longer the principal value proposition for employment of computers.

In contrast, Architecture is an *asset*. You *invest* in assets in order *to enable you to do something you otherwise are unable to do*. Assets are reusable, infrastructure. What makes an asset any different than an expense (a consumable)? It depends on how many times you want to use it. If you want to use it once, it is an expense. If you want to use it more than once, it is an asset. Architecture is an asset, infrastructure, an investment, *not* an expense. You cannot “cost-justify” Architecture because it is not displacing any costs. You do Architecture in order to do something you otherwise are unable to do. Oh ... it will save you inordinate amounts of time and money, not by displacing costs, but by enabling you to do four things you are unable to do without Architecture.

The Value Proposition for Architecture

Alignment.

The first reason you do Architecture is for “Alignment.” Do the implemented systems (Row 6) align with management’s intentions (Row 2) for the Enterprise?² Apparently, management does not perceive the systems to be “aligned” because in the last number of years, both CEO and CIO surveys rank “alignment” as one of the top several critical issues confronting I/S.

Here is how you use Architecture to achieve “Alignment:”

First, define the “Scope” of the Enterprise in the context of its external environment (Row 1). Next, define the conceptual “Models of the Business,” how Management intends to “use” the business that is, define their “Requirements” for the business (Row 2). Next, define the “Models of the Systems,” the logical “Specifications” for the business implementations (Row 3). Next, define the “Technology-Constrained, (physical) Models” of the business, the “Systems Design,” Row 4. Next, describe the “Systems Design” to the products that will be used for implementation (Row 5). Next, compile and run. Then, the implemented systems WILL align with the Business.

Sometimes I somewhat facetiously say, if you don’t like that approach to achieving “Alignment,” what’s YOUR approach?? It looks to me like any other approach would require either a suspension of the laws of nature or an inordinate amount of pure luck!

Furthermore, you will have to retain and maintain all of those models you produce in order to maintain the Alignment over time because over time, the technology products (Row 5) will change, the Systems Design (Row 4) will change, the Business

² The references to “Rows” and “Columns” and “Cells” in this article refer to the Framework for Enterprise Architecture. The website www.zifa.com contains many articles about as well as a picture of the Framework.

Specifications (Row 3) will change, the Management Requirements (Row 2) will change and the Enterprise Scope (Row 1) will change.

The quality criteria for any one cell will always be found in the cell above. For example, how do you know when you have quality Data Definition (Column 1, Row 5)? ... well, when the Data Definition accurately represents the Data Design (Column 1, Row 4). How do you know when you have quality Data Design (Column 1, Row 4)? ... well, when the Data Design (Column 1, Row 4) accurately represents the Logical Data Model (Column 1, Row 3). How do you know when you have a quality Logical Data Model (Column 1, Row 3)? ... well, when the Logical Data Model (Column 1, Row 3) accurately represents the Enterprise Semantic Model (Column 1, Row 2). How do you know when you have a quality Semantic Model (Column 1, Row 2)? ... well, when the Semantic Model (Column 1, Row 2) accurately represents the enterprise Scope (Column 1, Row 1). The quality of the end result, Row 6, in a total sense related to management's Scope and Requirements, Rows 1 & 2, is determined and maintained by incrementally controlling the quality cell by cell down any (every) Column.

In Manufacturing, the conceptual equivalent to "Alignment" is "Quality," "Total Quality Management (TQM)," that is, producing products (end results, Row 6) that meet the requirements and expectations of the customer (Rows 1 & 2).

If you do not have Architecture, there is NO WAY you are going to maintain "Alignment," let alone achieve it in the first place, short of an inordinate amount of pure luck.

Integration.

The second reason you do Architecture is "Integration."

I make a strong case in the seminars I do that in the Information Age, "Integration" is imperative *at least* in Column 1, "Data," the semantic structures of the Enterprise; Column 3, "Network," the connectivity of the Enterprise; and in Column 6, "Motivation," the ends/means of the Enterprise. Enterprise-wide, integrated Column 1 Models, Column 3 Models and Column 6 Models are the enablers of "Empowerment."

As soon as the same information is available to everyone in the Enterprise at the same time, the power will shift outboard in the Enterprise.³ No longer will all the power be concentrated in one or two people at the top who know everything, control everything, decide everything, etc. In fact, if the customer of the Enterprise has access to the same information at the same time that the Enterprise has access to it, the power will shift into the customer environment. The Enterprise will become "market-driven."

We see this phenomenon in the last 10 years' concentration on Data Warehouse, mostly around customer, the current emphasis in management seminars on "Customer

³ "Powershift" by Alvin Toffler. Bantam Books 1990.

Responsibility Management,” “One-on-One Marketing,” “The Customer is King,” etc., etc. As the “Powershift” materializes, so does the significance of “Empowerment.” The decision process must necessarily decentralize to the interface between the Enterprise and the customer. The people at the interface with the customer must be “empowered” to act quickly and decisively in response to the individual customer demand. In order to act quickly and decisively, the empowered people must be able to send messages to everybody else in the Enterprise (Column 3), those messages must mean the same thing or they might as well not bother to send the message (Column 1) and all those empowered people must have a clear idea what the motivation (Business Rules) of the Enterprise is (Column 6). Column 1 (Data), Column 3 (Network) and Column 6 (Business Rules) must be *integrated, Enterprise-wide*, in order to operate effectively in a customer-dominant marketplace.

In the Industrial Age, implementation took precedence over integration. It was more important to get the systems implemented, to displace expensive labor costs with machine costs than to protect enterprise-wide architectural continuity. We optimized the implementations at the expense of the data, the network and the business rules. Presently, the legacy data is a “tower of babel,” the data doesn’t mean the same thing to everyone and is useless for management purposes; the network is made up of every kind of hardware and systems software and versions known to mankind and is extremely fragile, rigid and costly to keep functioning; the business rules are incapable of being enforced consistently across the Enterprise to its detriment and (legal) liability; and change is virtually impossible to accommodate.

I would suggest that in the Information Age, integration must take precedence over implementation. The data **MUST** mean the same thing to everyone in the Enterprise if empowerment is to be successful, that is, if the Enterprise truly intends to realize Customer Relationship Management and to learn effectively (as an Enterprise) and function in a knowledge-based society. The hardware and systems software **MUST** be interchangeable if the network is to be functional 24 hours a day, 7 days a week, if it is to be maintained for less than half of the IT budget as currently surveyed, and if the Enterprise is to be maneuverable as new technologies pour into the marketplace. The business rules **MUST** be consistently understood and managed if the employees are to be empowered, the customers served, legal exposures minimized and the government satisfied. Enterprise-wide integration **MUST** be effected in Column 1, Column 3 and Column 6 at a minimum.

The Manufacturing equivalent to integration is “standard, interchangeable parts.” This is a “normalized” implementation. Components are shared (or, reused) rather than created uniquely for every employment.

If you have not produced Architecture, Enterprise-wide, there is **NO WAY** you are going to have normalized, shared, reusable components, that is, there is **NO WAY** you are going to achieve “Integration.”

Change.

The third reason you must have Architecture is for managing “Change.”

The descriptive representations (Architecture) of any object constitute the base-line for managing change to that object over time. How do you change buildings? You start with the drawings, functional specs, bills of material, etc., etc. How do you change airplanes, locomotives, computers, silicon chips, automobiles, lawnmowers, battleships, bicycles, DNA, etc., etc., etc? You start with the drawings, functional specs, bills of material, etc., etc. If you lose the drawings, functional specs, bills of material, etc., etc., that is, if you lose the diagrammatic representations that constitute Architecture, you have three alternatives.

It is easier to think about physical objects like airplanes, buildings, computers, etc. and see the implications of changing them without the architectural constructs. Therefore, take buildings as an illustration, in fact take a modest building, one that is only 20 stories or so. If you want to change it and you have lost the Architectural representations, here are your three alternatives:

1. Change it by trial and error. “Let’s just move this pillar out of the way to make the room a little more uncluttered.” This would be high risk. Who knows what is going to happen if you start moving building pillars around indiscriminately? In fact, no Architect or Contractor in their right mind is going to change even a modest 20 story building by trial and error!
2. Recreate (“reverse engineer”) the drawings, functional specs, bills of material, etc., etc. In fact, if an Architect or Contractor is going to try to change a building where they have no architectural representations, they will be in there with drills and tape measures recreating the models, probably up to at least the Row 3 level (“Designer’s View,” logical representations, “Architect’s Plans”) in order to understand the implications of changing it. They will go to the library and try to find newspaper articles and pictures of the building as it was being built. This takes time and costs money!
3. The third alternative is ... let the thing go obsolete, tear the whole thing down and start over again. This is the dinosaur option. Scrap the whole thing.

Let’s make this really simple. How do you change that PC on your desk? ... You start with the drawings, functional specs, bills of material, etc., the Architectural representations. What happens if you lose the Architectural representations and want to change it? ... You have three alternatives:

1. Change it by trial and error. Swap out a few jumpers on the motherboard and see what happens. Of course, this is okay as long as you don’t mind making it a boat anchor in the morning! High Risk.

2. Recreate (reverse engineer) the drawings, functional specs, bills of material. Sketch out the motherboard, locations of the pins, jumpers, etc., etc., then keep track of this functionality with that configuration, that functionality with this configuration, etc. etc. This takes time and costs money.
3. Let the thing go obsolete and just replace it with a new one. This is the “dinosaur option.” Throw this one out and start over again. Scrap the whole thing.

How do you change Enterprises if you have no drawings, functional specs, bills of material, etc? You have three options:

1. Change it by trial and error. Just change it and see what happens. High Risk!
2. Reverse engineer the “as is” logical data model (bill-of-materials, Column 1), the business process model (the functional specs, Column 2), the business logistics system (the drawings, Column 3), the work flow model (the operating instructions, Column 4), the dynamics model (the timing diagrams, Column 5), the objectives and strategies (the engineering decisions, Column 6). That takes time and costs money!
3. Don’t change it. Let it go obsolete (out of business) and start over again.

How do you change *anything*? You start with the drawings, functional specs, bills of material, etc., etc., the architectural representations (“as is” models) of the object you are going to change.

(Note: it would be A LOT better to actually engineer the object and keep the models rather than construct it by trial and error and then try to reverse engineer the models every time you want to change it!)

If you have no Architecture, there is NO WAY you are going to change anything with minimum time, disruption and cost.

Reduced “Time to Market.”

The fourth reason you do Architecture is to reduce the time it takes to produce an implementation from the time you receive the order.

The world is coming under extreme pressure to reduce “time-to-market.” No one is exempt. Enterprises are not exempt. I/S is not exempt. Individuals are not exempt. “Future Shock” has become a reality. As the rate of change continues to escalate, we are running out of time. No one has time to spare. If the present trend continues, it won’t be

long before, the time you have to produce a result from the time you get the order for it will approximate zero!

Toyota recently announced that if you sit down to a terminal and define the specs for an automobile, they will produce a custom automobile for you in five days! Several months ago I was in Singapore. There they told me that Toyota has been doing this in Japan for five years! Do you think that Toyota is waiting until they get your order before they start to engineer and manufacture the car? No way! They have the engineering and manufacturing done long before they ever get an order. Toyota is not making the cars to order, that is, making custom cars in a classic sense where they take the order first and then engineer and manufacture the car. They are not engineering and manufacturing standard cars to storage before they get the order in which case you would get a standard car off-the-shelf, not a custom car. What they are doing is engineering and manufacturing *parts* to storage that is, pre-fabricating the parts, so that when they get your order, all that they have to do is identify the parts required for your car, pick them off the shelf and assemble them into a custom car. The car is being assembled from a standard bill-of-materials of parts, where the *parts have been engineered to be assembled into more than one kind of car*. “Assemble-to-order.” “Mass customization.” They are engineering “*reusability*” into the parts. The time required to produce the car is only the time required to identify the parts required, pick them off the shelf and assemble them into the finished product ... 5 days. The engineering and manufacturing is done long before they get the order.

Just for comparison purposes, I recently saw some numbers that from concept (the time they get an idea for a new model) to customer (the time they deliver the first copy off the assembly line), one of the American automobile manufacturers required four YEARS! This is the “provide-from-stock” approach. The customer gets the product off-the-shelf (immediate delivery) but it is a *standard* product, not a custom product. The customer then has to change the use of the product to fit the product. (They DON'T change the product to fit the use!)

If you are a supplier (like I/S), and the customer (like the Enterprise) starts wanting *immediate delivery* on the systems you are engineering and manufacturing, but the customer (the Enterprise) is unable to define the characteristics of the systems they require until the moment they want to take delivery on them, how do you intend to satisfy their demand? Will you wait until you get the order until you start to engineer and manufacture the system? What is your time-to-market today? The average for major systems implementations I read somewhere is about 8 years. That is pretty good considering that major Enterprise systems are far more complex than automobiles! If you are “making-to-order,” the only way to reduce time-to-market is to reduce scope ... simplify. Simplify! Simplify!! Simplify!!! If you simplify to the extent that you can produce implementations in months, you are likely to be producing such simple things they are meaningless to today's complex, dynamic, information-dominant Enterprise.

One way you could reduce time-to-market to zero is to buy and install packages. That works as long as the Enterprise is able to change the Enterprise to fit the package. If the

Enterprise needs a custom implementation and you start changing the package to fit the Enterprise, it will likely take you a hundred times as long and cost a hundred times as much as making-it-to-order. You would be taking a standard product off-the-shelf and changing it into a different product. You would be better off engineering and manufacturing a custom product to order.⁴

The only way you will be able to reduce the application systems time-to-market to virtually zero is to do the engineering and manufacturing long before you get the order ... but you have to engineer and manufacture “parts,” (components) not finished goods (packages). That is, you have to engineer reusability into the components. That is, the models have to be engineered *Enterprise-wide*. The components have to be engineered such that they can be assembled into *more than one* implementation anywhere in the Enterprise. You have to have the components in inventory before you get the order but the components have to be engineered such that all you have to do is identify the appropriate components, pick them off the shelf and assemble them into the finished system. What do you think it is that you have in inventory before you get the order? Components engineered to be reused for any implementation in the Enterprise ... ARCHITECTURE!!

If you REALLY want to reduce time-to-market to virtually zero, you are going to have to have Architecture coupled with an assemble-to-order strategy. Conversely, if you REALLY want to reduce time-to-market for systems implementations by some orders of magnitude, there is NO WAY you are going to do this *without* “Architecture.”

Summary.

There are four reasons why you do Architecture: “Alignment,” “Integration,” “Change” and “Reduced Time-to-Market.”

Do you care that the systems I/S is producing actually are aligned with Management’s requirements and warrant the expenditure of funds allocated to I/S?

Do you care whether the data in the Enterprise means the same thing to anyone who uses it, that messages can be cost-effectively transmitted and received whatever time of day or night or day of the year, and that business rules can be uniformly enforced throughout the Enterprise?

Do you care whether changes to the Enterprise can be made with minimum time, disruption and cost?

Do you care whether I/S can produce “custom” implementations on demand, reducing their time-to-market to virtually zero?

⁴ “Packages Don’t Let You Off the Hook” by John A. Zachman. DataToKnowledge Newsletter, vol. 27, no. 4 (part 1) and no. 5 (part 2) July/August 1999 September/October 1999. Business Rule Solutions. 1-604-899-5452. www.brcommunity.com

If you care about any or all of these things, YOU ARE GOING TO DO ARCHITECTURE, because without Architecture, you cannot do ANY of these things.

The manufacturing equivalents for these characteristics are: Total Quality Management, Standard Interchangeable Parts, Change Management and Mass Customization. Nobody, but NOBODY in Manufacturing is arguing against Total Quality Management, Standard Interchangeable Parts, Change Management and Mass Customization any more. Tragically, many of us in I/S can find all the reasons why you can't do Architecture.⁵ (It was the *gunsmiths themselves* that argued against standard interchangeable parts and mass production for rifles during the Civil war!)

You can't "cost-justify" Architecture. Architecture is not an expense. Architecture does not displace any other costs. Architecture is an asset. You can save orders of magnitude more money and time, but you have to *invest* in Architecture to enable you to do something you otherwise are unable to do, namely: "Alignment," "Integration," "Change" and "Mass Customization."

That is the value proposition for Architecture.

Architecture is an Information Age idea. "Cost-justification" was an Industrial Age idea.

Note: Some day, I guarantee you, you are going to wish you had every one of the models identified by the Framework for Enterprise Architecture made explicit, every one of them made explicit Enterprise-wide, all the models integrated horizontally across every Row, all the models integrated vertically down every Column and all the models made explicit at excruciating levels of detail. Forget you! Someday, the ENTERPRISE is going to wish they had all the models made explicit, Enterprise-wide, horizontally and vertically integrated at excruciating levels of detail. It is inevitable! This is the Information Age!! The question is NOT, "is this a good idea?" Or, "is this what you really want to do?" This is going to be what you HAVE TO DO if you want to play in the game!! The question is, "How do you intend to actually accomplish it over some period of time with minimum 'scrap and rework'?" "How do you get there as soon as possible with the least investment of money and time ... before anybody else gets there?!" That is, "how do you intend to build it out Column by Column, Row by Row, Cell by Cell, *sliver of cell by sliver of cell*, satisfying current demand but at the same time minimizing the architectural discontinuities that constitute the 'legacy' of today, which is requiring *maximum* 'scrap and rework'?" There are ways to do this! But if you don't get a pretty clear idea about, and a very strong commitment to achieving a coherent, integrated, holistic, aligned,

⁵ "All the Reasons Why You Can't Do Architecture: We Has Met the Enemy and He Is Us." John A. Zachman DataToKnowledge Newsletter, Vol. 28, No. 4 and 5, July/August 2000 and September/October 2000. www.brcommunity.com

flexible, adaptable, dynamic, maneuverable, reliable, efficient, effective *Enterprise* and start working on making it a reality, you are *never* going to get there. You won't even get close. And "... someday ... you are going to wish ... forget you ... someday, THE ENTERPRISE is going to wish it had all these models etc., etc., etc."

John A. Zachman
Zachman International
2222 Foothill Blvd. Suite 337
La Cañada, Ca. 91011
818-244-3763 (Phone and Fax)
johnzachman@compuserve.com