PERTH, AUSTRALIA – December 23, 2003: Earlier issues of TEN have separately addressed concepts of Enterprise Architecture and also of XML and Web Services for Enterprise Integration. This issue we will discuss changes in Systems Development methods that are necessary for evolution into the 21st Century Enterprise.

Clive Finkelstein
TEN - The Enterprise Newsletter

Back to Contents.
Enterprise Architecture and Enterprise Engineering achieve Business Integration in the enterprise for more effective Technology Integration. Before we examine these further, we first need to review the evolution of Systems Development Methodologies in the Information Age.

Methodologies evolved from the start of the Information Age to help us examine current manual processes so we could automate them. From rudimentary methodologies in the 1960s, by the 1970s these had evolved into the Software Engineering methods. Michael Jackson, Ken Orr, Ed Yourdon, Tom de Marco and others were key originators of Software Engineering methodologies: also called Structured Methods. These methods analyzed manual processes, documenting them with Data Flow Diagrams (DFDs) and Functional Decomposition Diagrams (FDDs). The structure of modular programs to automate these processes was documented using Structure Charts (SCs). Programs were then written in various programming languages to execute the automated processes.

As discussed in the last issue of TEN, in automating manual processes AS-IS, we moved from manual chaos to automated chaos. This was due to the fact that common manual processes, used in various parts of the enterprise, had often evolved in quite different ways. For example, a process to manually accept an order (an Order Entry Process) may be carried out differently according to how the order was received: by mail; or by phone; or instead from a salesperson. The Order Entry Process may also be carried out differently depending on the specific products or services that were ordered. The result was the evolution of different manual processes, all intended to achieve the same objective: Accept an Order for Processing. When these manual processes were automated, we found we also had many automated Order Entry Processes, all intended to Accept an Order for Processing. We had lost sight of the principle of Reusability, first identified by Adam Smith in his book, “Wealth of Nations”, published in 1776 and discussed last issue.

This added to the automated chaos: when a change had to be made to a process, the same change had to be made to every version of that process throughout the enterprise. Every program that automated the different versions of the process had to be changed, often in slightly different ways. The result was also chaos: Program Maintenance Chaos!
With Software Engineering, each DFD that was defined for a process identified the data that it needed: as Data Stores. Each different version of the same process resulted in a different data store version of required data. Redundant data versions were implemented for each automated process. In fact, we earlier saw this problem start to emerge with the 19th Century industrial enterprise: we discussed last issue that Insurance and Bank workers each kept a separate hand-written Applicant Ledger or Customer Ledger to answer queries. What was the result?

*With these redundant data versions, we moved to Data Maintenance Chaos!*

Whenever a change had to be made to data values for maintenance purposes, such as by changing a customer’s address, every version of that address had to be changed. *This was redundant data maintenance processing. It required redundant staffing to do this redundant work.*

And because redundant data maintenance programs were developed independently, these data maintenance workers also had to be trained in the different operating procedures that were used for data entry by each redundant data maintenance program. *This resulted in redundant training.*

This approach to systems development has lead to the major problems that we have today of redundant data and processes with stovepipe systems. All of these redundant costs are regularly incurred by every enterprise today: in redundant data maintenance costs for redundant data value changes; and in redundant staffing and redundant training to carry out this redundant work.

*These are all redundant costs. They negatively affect the bottom line – in reduced profits for Commercial enterprises, or in reduced cost-effectiveness for Government or Defense enterprises.*

In this same period – from the late 60s through the early 70s, Edgar Codd (recently deceased) was a Research Fellow at IBM San Jose Labs where he developed the Relational Model from set theory. This was the foundation of the Relational Data Base technology that we use today. The first Relational Data Base Management (RDBMS) systems were released by IBM Corporation (IBM DB2 RDBMS) and by Oracle Corporation (Oracle RDBMS) in the late 70s and early 80s.
From the mid 70s, three approaches emerged to apply concepts of the Relational Model to the methods that were used for Data Base Design. One approach was developed in UK and Europe; another approach was developed in the USA; and a third approach was developed independently in Australia. Each addressed development of Data Modelling methods, using Normalization to eliminate redundant data versions.

The Australian approach also evolved into integrated methods for information, using a rigorous engineering discipline, called: Information Engineering (IE). Originally developed from 1976 by Clive Finkelstein, IE was popularized world-wide throughout the 1980s by James Martin. Further books showed the use of Business-Driven Information Engineering. This latter IE variant evolved into what is today called: Enterprise Engineering. We will now review the Systems Development strategies that are used today.

Back to Contents.

Systems Development Strategies Used Today

The approach that is used to design and build enterprise systems with traditional systems development methods is summarized below:

- Systems Requirements have typically been defined by IT staff, by interviewing users to determine their operational business needs.
- The designs that are established are then based on Technology, with Application Design, Database Design and Object Design reflecting that technology.
- The technology is used to deliver the desired business benefits and designs are then implemented to meet desired business Performance Requirements.

This approach to Systems Development is *Technology-Dependent*, and has resulted in problems:

- The Business Needs have been difficult to determine. If these needs are not understood or expressed clearly, the designed systems may not address the real needs of the users and management.
- The systems that are developed are typically not aligned with corporate goals that set the directions for the future. This is
one of the main problems with systems development today.

- But the strategic directions are not clear; yet they must be understood if IT is to design flexible systems that support the strategic directions.

In fact, problems with traditional development methods are much greater than suggested above. The business needs have traditionally been decided by reviewing the operational processes of the business. These processes were determined based on strategic plans typically defined many years ago, sometimes even a decade ago.

Yet in the early 90s we had no idea – not even in our wildest predictions of the future – that we would today be able to communicate instantly with customers, suppliers and business partners anywhere in the world, through the Internet. The environment that we accept today as the norm was way beyond our most fanciful imagination. Fact is sometimes stranger than fiction.

The strategic plans defined in the 90s did not anticipate that these organizations would today communicate with each other in seconds. They assumed communication would be as it had always been, by mail – or later by fax – with responses received days or weeks later. The most rapid response these business processes assumed was at best in hours. The business processes we still use today were never designed to respond in seconds.

This point is critical: The traditional systems development approach – interviewing users based on existing business processes and then identifying their future needs – does not work well in periods of rapid change, such as today.

In fact I will make this point stronger: If we base our needs for the future on operational processes that we still use today – we are implicitly assuming that the future will be similar to the past. This is dangerous; very few industries and enterprises can say today that their future will be like their past. Most know that the future will be different. The only certainty we have is that the processes we need then will be quite different from the processes we use today. This brings us to a very important principle for change:

We must design for tomorrow based not on operational processes still used today. We have to design for tomorrow by using new reusable activities and processes that are tailored for the environment of the
Internet – which represents our present and our future – so that enterprises can respond in seconds or minutes, not in days or weeks.

We must design for a future where the only thing that remains constant – is change itself.

Businesses must change, to compete with other organizations in their relevant markets. This is certainly true for Commercial organizations, which compete with other organizations. It is true for Government Departments that compete with other departments for government funding. And it is also true for Defense, which competes with hostile Defense forces, and also with friendly Defense forces for limited resources.

Competition today demands systems that can change easily to support rapid business change. Many of these business changes may need significant change or redevelopment of systems. Yet most of those systems were not designed for change. Existing systems may need massive modification to support essential business changes. Often it is faster to throw existing systems away and start over again: developing new systems from scratch. This is slow and very costly.

The advantages and benefits of Technology were not clear in the early 90s to many senior managers. It was sometimes difficult to get funding approval for new projects and funding for the resources that are vital for success. But the Internet and the Year 2000 problem in the late 90s both demonstrated to management the dramatic impact – both positive and negative – that Technology can have on the enterprise.

We have taken a bottom-up view with traditional methods in building systems for the enterprise. We looked at the existing systems – whether manual or automated. From this bottom-up view, we looked at ways in which current manual or automated systems have been implemented. We then examined ways to improve these systems: either by automating manual systems; or by using technology to improve existing automated systems.

By designing for tomorrow based on the existing business processes of today, we are basing our designs for the future on plans set perhaps a decade ago. Those plans never contemplated the rapid change environment that we operate under, today. And we already know that the business processes we have today do not enable the
business to change rapidly. So the key questions for the future that present themselves to us today are these:

- How can we design for a future where we will need to be able to change rapidly – and often?
- How can we address these problems, while also involving senior managers and business experts who set directions for that future?

Back to Contents.

Systems Development Strategies for Tomorrow

Solutions to the above questions – so our enterprises can respond effectively in an environment of rapid business change – must address the following imperatives:

- Systems that are developed for the future must support the corporate goals. This is the most common systems development problem today.
- We must therefore determine the goals for the future. But goals are expressed in business terms, not in systems terms. How can we determine what to implement?
- IT Departments must be aware of strategic directions so they can design for the future. This is difficult, as most IT Departments do not participate in Strategic Business Planning.
- IT Departments must build systems based on strategic business plans if those systems are to be aligned with corporate goals. They must be based on activities and processes designed for the future, not the past.

If this is done, Technology can offer competitive advantage: it can be used to help achieve the strategic business plans and corporate goals, with new activities and processes that respond in seconds or minutes – not in days or weeks.

Enterprise Engineering resolves many of these problems with Systems Development today. It enables Business Experts and IT Experts to work together in a Design Partnership – using Enterprise Engineering driven by Strategic Business Plans that set directions for the future.
Enterprise Engineering supports: **Enterprise Architecture; Business Re-Engineering; Forward Engineering; and Reverse Engineering.** These business-driven methods are used to:

- Build Systems for the future that can support the corporate goals.
- Identify goals for the future in business terms, so that IT can determine what to implement in systems terms.

Enterprise Engineering provides strategic business planning methods so that the IT Department can participate in strategic business planning with management. It enables IT to build systems based on Strategic Plans so that those systems are aligned with corporate goals. Technology can then offer competitive advantage – used to help achieve the strategic plans and corporate goals.

We will now examine the Business-Driven Enterprise Engineering methodologies in more detail. These methods support all phases of the Systems Development Life Cycle (SDLC). **Figure 1** illustrates that Phases above the line are **Technology-Independent** methods and focus on the business. They apply to Rows 1 – 3 (Planner, Owner and Designer) of the Zachman Framework for Enterprise Architecture. These methods are **Strategic Business Planning, Data Modelling and Function Modelling:**

- The Strategic Directions set by management provide input to Strategic Business Planning. These are addressed in Col 6 (Why) of the Zachman Framework.
- These plans indicate the Information Requirements of management that are input to Data Modelling in Col 1 (Data): Strategic; Tactical; and Operational Data Modelling.
- Concurrently, plans and data models indicate how Information Usage is input to Function Modelling: for Activity Modelling; Scenario Modelling; and Process Modelling in Col 2 (How).
Figure 1: Enterprise Engineering supports the Zachman Framework for Enterprise Architecture, with rapid implementation of priority Enterprise Architecture areas

The above phases of Figure 1 define Technology-Independent business requirements and address Enterprise Architecture Rows 1-3. Phases below the line in Figure 1 are Technology-Dependent and address the Enterprise Architecture Rows 3-5 (for Designer, Builder and Subcontractor). These methods address Component Design and Systems Implementation:

- Technology and Systems Requirements provide input to Systems Design. Web Services and Service-Oriented Architecture (SOA) XML-based Business Process Management (BPM) and Object-Oriented methods in this phase are used for Application Design, Data Base Design and Object Design of systems to be deployed on corporate Intranets and/or the Internet.
- Identified Performance Requirements then provide the input required by the Systems Implementation phase.

The result of these Enterprise Engineering methods for the 21st Century Enterprise is the rapid identification of reusable business activities and business processes that can be implemented once, yet shared many times enterprise-wide. Redundant data versions and their consequent redundant processes are eliminated. Business objects are implemented once only, yet shared enterprise-wide.
These are the business objects whose identification has been so elusive, when approached traditionally using object-oriented methods from a bottom-up perspective.

The same object-oriented methods are now able to implement these business objects rapidly as reusable business processes, with reusable data and methods. Changes can be applied rapidly; once changes have been made to relevant business objects, all systems in the enterprise that share those same business objects immediate operate using these changes.

Alternatively, Business Process Management (BPM) languages used for Service-Oriented Architecture (SOA) can be automatically generated as executable XML-based code directly from Workflow Models. These BPM languages can also be automatically generated now directly from UML diagrams. Two examples are: executable Business Process Specification Schema (BPSS) code, as defined for ebXML; and Business Process Execution Language for Web Services (BPEL4WS, or just BPEL) derived from UML diagrams, as developed now by IBM following its purchase of Rational. Another example is automatic generation of executable BPEL code by Microsoft from BizTalk Process Orchestration diagrams using BizTalk Server 2004.

The overall result of using Enterprise Engineering with Enterprise Architecture – building systems based on strategic plans for the future as described above – is dramatic savings in development time and cost. Redundant data maintenance costs are eliminated, with further large cost savings in operational processing experienced with today’s stovepipe systems. And the business objects that have been implemented once, yet shared enterprise-wide, permit rapid business change – in days and sometimes hours, no longer in months or years as we have with today’s stovepipe systems.

Back to Contents.

---

Summary of Systems Development Strategies

I will review the overall messages for the future, drawn from previous TEN issues, for the underlying arguments that support this brief summary.

- Adam Smith’s book, “Wealth of Nations” (1776), was influential
for the Industrial Age. It described the evolution from the Agricultural Age to the Industrial Age. It was the foundation for most industrial enterprises in the late 18th Century and into the 19th Century.

- By the middle of the 20th Century, the Industrial Enterprise had evolved into a complex series of manual processes. The pace of progress had seen most enterprises evolve to use increasingly complex business processes, with rapidly growing transaction volumes to be manually processed. These enterprises found … they were operating in a continual state of Manual Chaos!

- From the late 1950s, manual processes were automated by computer. We took the existing manual processes and then automated them essentially AS-IS: without much change. In so doing, we moved from Manual Chaos … instead to Automated Chaos!

- With rapid acceptance of the Internet in the second half of the 90s the chaos moved from the back office … onto the front doorstep of enterprises: through their web sites. We saw that customers could visit these enterprises by the click of a mouse. But they could just as quickly leave … if the processes did not provide what the customers needed.

- The problem is that we have 21st Century Enterprises that use 21st Century Technologies … yet most enterprises today still use 18th Century Disintegrated Business Processes!

- The business processes – originally designed based on principles set by Adam Smith in 1776 – have not evolved to take advantage of the technologies we have today. We need integrated 21st Century Enterprises together with integrated 21st Century Processes!

- We discussed the problem of redundant data versions in most enterprises. When data values change, all redundant versions must be updated to synchronize with that change. But with redundant data, we moved to Data Maintenance Chaos!

- We also discussed evolution of Systems Development Methodologies: Software Engineering; Information Engineering; and Object-Oriented methods.

- We saw that in spite of these methods, process changes that require procedural program changes resulted in Program Maintenance Chaos! Our procedural programming methods were not designed so they could accommodate change easily, and object-oriented methods could not identify enterprise-wide reusable code.

- We saw that many Data Maintenance and Program Maintenance problems are resolved by Business Integration. We learned that Business Integration is best achieved by using Enterprise Architecture.
● We discussed Enterprise Architecture and the concepts of the Zachman Framework for Enterprise Architecture. We saw that the true architects of an enterprise are not in IT:

● Enterprise Architects are the senior managers who set the directions for the future, based on business plans, strategies and processes for that future and its technologies.

● The future will be based on business processes that use the technologies of today and tomorrow ... with strategic directions set by senior management.

● From these strategic directions, business experts and IT experts can then work together in a design partnership to address these needs of the future.

● We discussed the concepts of Enterprise Engineering for rapid definition and delivery of Enterprise Architecture.

● When the Zachman Framework for Enterprise Architecture is used from the perspectives of the Planner and the Owner rows, with Enterprise Engineering applied enterprise-wide, the reusable activities and processes within an enterprise can be readily identified.

The key messages that I want to leave you with – for evolution to the 21st Century Enterprise – are therefore these:

● Rather than continue to build systems based on operational processes that reflect the needs of the past, by basing our designs for the future on the business plans that define that future we can build systems that can be implemented rapidly and changed easily.

● We must design for tomorrow based on business plans for the future. Our designs must draw on the knowledge of senior management and their business experts, reflected in the strategic business plans of the enterprise.

● We must use activities and processes that have been defined enterprise-wide by business experts applying these plans throughout the enterprise, to identify reusable activities and processes.

● From this enterprise-wide perspective, we can implement these reusable business activities and processes as business objects that can be implemented once, yet shared many times.

● Once implemented, these systems can respond to rapid business change environments as we have today. We will no longer be tied to inflexible stovepipe systems that cannot be changed easily.

● We can build for this future using Enterprise Architecture and Enterprise Engineering, together with Object-Oriented
methods and technologies such as Web Services and Service-Oriented Architecture (SOA) Business Process Management (BPM) XML-based languages: to implement in weeks or days what previously took years or months.

Rather than 18th Century processes that we have today in most enterprises, the end-result will be integrated 21st Century Enterprises, together with integrated 21st Century Processes that can be implemented easily, yet changed rapidly and often.

Back to Contents.

AUTHOR

Clive Finkelstein is the "Father" of Information Engineering (IE), developed by him from 1976. He is an International Consultant and Instructor, and Managing Director of Information Engineering Services Pty Ltd (IES) in Australia.


For More Information, Contact:

Clive Finkelstein, Managing Director
Information Engineering Services Pty Ltd
PO Box 246, Hillarys, Perth WA 6923 Australia

Details: http://www.ies.aust.com/cbfindex.htm
Web Site: http://www.ies.aust.com/
Phone: +61-8-9402-8300
Fax: +61-8-9402-8322
Email: cfink@ies.aust.com