TITLE: "A SHORT HISTORY OF SYSTEMS DEVELOPMENT"

by Tim Bryce
Managing Director
M. Bryce & Associates (MBA)
P.O. Box 1637
Palm Harbor, FL 34682-1637
United States
Tel: 727/786-4567
E-Mail: timb001@phmainstreet.com
WWW: http://www.phmainstreet.com/mba/
Since 1971: *"Software for the finest computer - the Mind"*

"If they do not have an appreciation of whence we came, I doubt they will have an appreciation of where we should be going."
- Bryce's Law

INTRODUCTION

I always find it amusing when I tell a young person in this industry that I worked with punch cards and plastic templates years ago. Its kind of the same dumbfounded look I get from my kids when I tell them we used to watch black and white television with three channels, no remote control, and station signoffs at midnight. It has been my observation that our younger workers do not have a sense of history; this is particularly apparent in the systems world. If they do not have an appreciation of whence we came, I doubt they will have an appreciation of where we should be going. Consequently, I have assembled the following chronology of events in the hopes this will provide some insight as to how the systems industry has evolved to its current state.

I'm sure I could turn this into a lengthy dissertation but, instead, I will try to be brief and to the point. Further, the following will have little concern for academic developments but rather how systems have been implemented in practice in the corporate world.

PRE-1950'S - "SYSTEMS AND PROCEDURES"

Perhaps the biggest revelation to our younger readers regarding this period will be that there was any form of systems prior to the advent of the computer. In fact, "Systems and Procedures" Departments predated the computer by several years. Such departments would be concerned with the design of major business processes using "work measurement" and "work simplification" techniques as derived from Industrial Engineering. Such processes were carefully designed using grid diagrams and flowcharts. There was great precision in the design of forms to record data, filing systems to manage paperwork, and the use of summary reports to act as control points in systems. For example, spreadsheets have been extensively used for many years prior to the introduction of Lotus 1-2-3 or MS Excel. There was also considerable attention given to human behavior during the business process (the precursor to "ergonomics").

Systems were initially implemented by paper and pencil using ledgers, journals (logs), indexes, and spreadsheets. We have always had some interesting filing systems, everything from cards and folders, to storage cabinets.

Perhaps the earliest mechanical device was the ancient abacus used for simple math (which is still used even to this day). The late 1800's saw the advent of cash registers and adding machines as popularized by such companies as NCR in Dayton, Ohio under John Patterson who also introduced sweeping changes in terms of dress and business conduct. This was adopted by Thomas Watson, Sr. who worked for many years at NCR and carried forward these practices to IBM and the rest of the corporate world. Also, Burroughs was a major player in the early adding machine industry.

The first typewriters were also introduced in the late 1800's which had a tremendous effect on correspondence and order processing. This was led primarily by Remington Arms (later to become Remington Rand).

In the early 1900's, tabulating equipment was introduced to support such things as census counting. This was then widely adopted by corporate America. Occasionally you will run into old-timers who can describe how they could program such machines using plug boards. Punch card sorters were added as an adjunct to tabulating equipment.

As a footnote, most of what IBM's Watson learned about business was from his early days at NCR. However, he had a falling out with Patterson who fired him. As a small bit of trivia, after Watson died, he was buried in Dayton on a hilltop overlooking NCR headquarters, the company he couldn't conquer.

During World War II, both the U.S. military and industrial complex relied heavily on manually implemented systems. We did it so well that many people, including the Japanese, contend it gave the Allies a competitive edge during the war.

The lesson here, therefore, is that manually implemented

(continued on page 2)

(continued from page 1)

systems have been with us long before the computer and are still with us today. To give you a sense of history in this regard, consider one of our more popular Bryce's Laws:

"The first on-line, real-time, interactive, data base system was double-entry bookkeeping which was developed by the merchants of Venice in 1200 A.D."

One major development in this area was the work of Leslie "Les" Matthies, the legendary Dean of Systems. Les graduated from the University of California at Berkeley during the Depression with a degree in Journalism. Being a writer, he tried his hand at writing Broadway plays. But work was hard to come by during this period and when World War II broke out, Les was recruited by an aircraft manufacturer in the midwest to systematize the production of aircraft. Relying on his experience as a writer, he devised the "Playscript" technique for writing procedures. Basically, Les wrote a procedure like a script to a play; there was a section to identify the procedure along with its purpose; a "Setup" section to identify the forms and files to be used during it; and an "Operations/ Instructions" section which described the "actors" to perform the tasks using verbs and nouns to properly state each operation. He even went so far as to devise rules for writing "If" statements.

For details on "Playscript," see "PRIDE" Special Subject Bulletin No. 38 - *"The Language of Systems"* - Aug. 22, 2005
http://www.phmainstreet.com/mba/ss050822.pdf

"Playscript" became a powerful procedure writing language and was used extensively throughout the world. It is still an excellent way to write procedures today. Ironically, Les did not know what a profound effect his technique would have later on in the development of computer programs.

1950'S - INTRODUCTION OF THE COMPUTER

Yes, I am aware that the ENIAC was developed for the military at the end of World War II. More importantly, the UNIVAC I (UNIVversal Automatic Computer) was introduced in 1951 by J. Presper Eckert and John Mauchly. The UNIVAC I was a mammoth machine that was originally developed for the U.S. Bureau of the Census. Corporate America took notice of the computer and companies such as DuPont in Wilmington, Delaware began to lineup to experiment with it for commercial purposes. The Remington Rand Corporation sponsored the project, but the company's focus and name eventually changed to "UNIVAC" (today it is referred to as "UNISYS," representing a merger of UNIVAC with Burroughs).

The UNIVAC I offered a sophistication unmatched by other manufacturers, most notably IBM's Mach I tabulating equipment. This caused IBM to invent the 701 and its 700 series. Other manufacturers quickly joined the fray and computing began to proliferate. Although UNIVAC was the pioneer in this regard, they quickly lost market share due to the marketing muscle of IBM. For quite some time the industry was referred to as "IBM & the BUNCH" (Burroughs, UNIVAC, NCR, CDC, and Honeywell).

Programming the early machines was difficult as it was performed in a seemingly cryptic Machine Language (the first generation language). This eventually gave way to the Assembly Language (the second generation language) which was easier to read and understand. Regardless, many of the utilities we take for granted today (e.g., sorts and merges) simply were not available and had to be developed. In other words, programming was a laborious task during this period.

Recognizing both the limitations and potential of the computer, the 1950's represented the age of experimentation for corporate America. Here, the emphasis was not on implementing major systems through the computer, but rather to develop an assortment of programs to test the machine as a viable product. As such, programmers were considered odd characters who maintained "the black box," and were not yet considered a part of the mainstream of systems development. The "Systems and Procedures Departments" still represented the lion's share of systems work in corporate America, with an occasional foray to investigate the use of the computer.

The computer people were segregated into "computer departments" (later to be known as "EDP" or "Data Processing" departments).

1960's - MANAGEMENT INFORMATION SYSTEMS

Competition between computer manufacturers heated up during this decade, resulting in improvements in speed, capacity, and capabilities. Of importance here was the introduction of the much touted IBM 360 (the number was selected to denote it was a comprehensive solution - 360 degrees). Other computer vendors offered products with comparable performance, if not more so, but the IBM 360 was widely adopted by corporate America.

The programming of computers was still a difficult task and, consequentially, Procedural Languages were intro-

(continued on page 3)

(continued from page 2)

duced (the third generation languages). In actuality, these languages got their start in the late 1950's, but the proliferation of computers in the 1960's triggered the adoption of procedural languages such as COBOL, FORTRAN, and PL/1. Interestingly, these languages were patterned after Les Matthies' "Playscript" technique which made active use of verbs, nouns, and "if" statements.

The intent of the Procedural Languages was twofold: to simplify programming by using more English-like languages, and; to create universal languages that would cross hardware boundaries. The first goal was achieved, the second was not. If the languages were truly universal, it would mean that software would be portable across all hardware configurations. Manufacturers saw this as a threat; making software truly portable made the selection of hardware irrelevant and, conceivably, customers could migrate away from computer vendors. In order to avoid this, small nuances were introduced to the compilers for the Procedural Languages thereby negating the concept of portability. This issue would be ignored for many years until the advent of the Java programming language.

The 1960's also saw the introduction of the Data Base Management System (DBMS). Such products were originally designed as file access methods for Bill of Materials Processing (BOMP) as used in manufacturing. The "DBMS" designation actually came afterwards. Early pioneers in this area included Charlie Bachman of G.E. with his Integrated Data Store (IDS) which primarily operated under Honeywell GCOS configurations; Tom Richley of Cincom Systems developed TOTAL for Champion Paper, and; IBM's BOMP and DBOMP products. In 1969, IBM introduced IMS which became their flagship DBMS product for several years.

With the exception of IMS, the early DBMS offerings were based on a network model which performed chain-processing. IMS, on the other hand, was a hierarchical model involving tree-processing.

Realizing that programming and data access was becoming easier and computer performance being enhanced, companies now wanted to capitalize on this technology. As a result, corporate America embarked on the era of "Management Information Systems" (MIS) which were large systems aimed at automating business processes across the enterprise. These were major system development efforts that challenged both management and technical expertise.

It was the MIS that married "Systems and Procedures" departments with computing/EDP departments and transformed the combined organization into the "MIS" department. This was a major milestone in the history of systems. The systems people had to learn about computer technology and the programmers had to learn about business systems.

Recognizing that common data elements were used to produce the various reports produced from an MIS, it started to become obvious that data should be shared and reused in order to eliminate redundancy, and to promote system integration and consistent data results. Consequently, Data Management (DM) organizations were started, the first being the Quaker Oats Company in Chicago, Illinois in 1965. The original DM organizations were patterned after Inventory Control Departments where the various components were uniquely identified, shared and cross-referenced. To assist in this regard, such organizations made use of the emerging DBMS technology. Unfortunately, many DM organizations lost sight of their original charter and, instead, became obsessed with the DBMS. Data as used and maintained outside of the computer was erroneously considered irrelevant. Even worse, the DBMS was used as nothing more than an elegant access method by programmers. Consequently, data redundancy plagued systems almost immediately and the opportunity to share and reuse data was lost. This is a serious problem that persists in companies to this day.

1970's - AWAKENING

Although the MIS movement was noble and ambitious in intent, it floundered due to the size and complexity of the task at hand. Many MIS projects suffered from false starts and botched implementations. This resulted in a period where a series of new methods, tools and techniques were introduced to reign in these huge development efforts.

The first was the introduction of the "methodology" which provided a road map or handbook on how to successfully implement systems development projects. This was pioneered by MBA with its "PRIDE" methodology in 1971. Although the forte of "PRIDE" was how to build systems, it was initially used for nothing more than documentation and as a means to manage projects. Following "PRIDE" was John Toellner's Spectrum I methodology and SDM/70 from Atlantic Software. Several CPA based methodologies followed thereafter.

Also during this time, mainframe based Project Management Systems were coming into vogue including Nichols N5500, PAC from International Systems, and PC/70 from Atlantic Software.

(continued on page 4)

(continued from page 3)

The early methodologies and Project Management Systems give evidence of the orientation of systems departments of that time: a heavy emphasis on Project Management. Unfortunately, it was a fallacy that Project Management was the problem; instead people simply didn't know how to design and build systems in a uniform manner. As companies eventually learned, Project Management is useless without a clear road map for how to build something.

In the mid-to-late 1970's several papers and books were published on how to productively design software thus marking the beginning of the "Structured Programming" movement. This was a large body of work that included such programming luminaries as Barry Boehm, Frederick P. Brooks, Larry Constantine, Tom DeMarco, Edsger Dijkstra, Chris Gane, Michael A. Jackson, Donald E. Knuth, Glenford J. Myers , Trish Sarson, Jean Dominique Warnier, Generald M. Weinberg, Ed Yourdon, as well as many others. Although their techniques were found useful for developing software, it led to confusion in the field differentiating between systems and software. To many, they were synonymous. In reality, they are not. Software is subordinate to systems, but the growing emphasis on programming was causing a change in perspective.

The only way systems communicate internally or externally to other systems is through shared data; it is the cohesive bond that holds systems (and software) together. This resulted in the introduction of Data Dictionary technology. Again, this was pioneered by MBA with its "PRIDE" methodology (which included a manually implemented Data Dictionary) and later with its "PRIDE"-LOGIK product in 1974. This was followed by Synergetics' Data Catalogue, Data Manager from Management Software Products (MSP), and Lexicon by Arthur Andersen & Company.

The intent of the Data Dictionaries was to uniquely identify and track where data was used in a company's systems. They included features for maintaining documentation, impact analysis (to allow the studying of a proposed change), and redundancy checks. "PRIDE"-LOGIK had the added nuance of cataloging all of the systems components, thereby making it an invaluable aid for design and documentation purposes.

The Data Dictionary was also a valuable tool for controlling DBMS products and, as such, several adjunct products were introduced, such as UCC-10, DB/DC Data Dictionary, and the Integrated Data Dictionary (IDD) from Cullinet. Unlike the other general purpose Data Dictionaries, these products were limited to the confines of the DBMS and didn't effectively track data outside of their scope.

DBMS packages proliferated during this period with many new products being introduced including ADABAS, Image, Model 204, and IDMS from Cullinet (which was originally produced at BF Goodrich). All were based on the network-model for file access which was finally adopted as an industry standard (CODASYl).

There were a few other notable innovations introduced, including IBM's Business Systems Planning (BSP) which attempted to devise a plan for the types of systems a company needed to operate. Several other comparable offerings were introduced shortly thereafter. Interestingly, many companies invested heavily in developing such systems plans, yet very few actually implemented them.

Program Generators were also introduced during this period. This included report writers that could interpret data and became a natural part of the repertoire of DBMS products. It also included products that could generate program source code (COBOL predominantly) from specifications. This included such products as System-80 (Phoenix Systems), GENASYS (Generation Sciences), and JASPOL (J-Sys of Japan), to mention but a few.

MBA also introduced a generator of its own in 1979 - a Systems generator initially named ADF (Automated Design Facility) which could automatically design whole systems, complete with an integrated data base. Based on information requirements submitted by a Systems Analyst, ADF interacted with the "PRIDE"-LOGIK Data Dictionary to design new systems and, where appropriate modify existing systems. Because of its link to LOGIK, ADF emphasized the need to share and reuse information resources. Not only was it useful as a design tool but it was a convenient tool for documenting existing systems. The only drawback to ADF was that the mindset of the industry was shifting from systems to software. Consequently, program generators captured the imagination of the industry as opposed to ADF.

The increase in computer horsepower, coupled with new programming tools and techniques, caused a shift in perspective in MIS organizations. Now, such departments became dominated by programmers, not systems people. It was here that the job titles "Systems Analyst" and "Programmer" were married to form a new title of "Programmer/Analyst" with the emphasis being on programming and not on front-end systems design. Many managers falsely believed that developers were not being productive unless they were programming. Instead of *"Ready,*

(continued on page 5)

(continued from page 4)

Aim, Fire," the trend became *"Fire, Aim, Ready."*

Data Management organizations floundered during this period with the exception of Data Base Administrators (DBA's) who were considered the handmaidens of the DBMS.

The proliferation of software during this decade was so great that it gave rise to the packaged software industry. This went far beyond computer utilities and programming tools. It included whole systems for banking, insurance and manufacturing. As a result, companies were inclined to purchase and install these systems as opposed to re-inventing the wheel. Among their drawbacks though was that they normally required tailoring to satisfy the customer's needs which represented modification to the program source code. Further, the customer's data requirements had to be considered to assure there were no conflicts in how the customer used and assigned data. After the package had been installed, the customer was faced with the ongoing problem of modifying and enhancing the system to suit their ever-changing needs.

1980's - THE TOOL-ORIENTED APPROACH

As big iron grew during the 1960's and 1970's, computer manufacturers identified the need for smaller computers to be used by small to medium-sized businesses. In the 1970's, people were skeptical of their usefulness but by the 1980's their power and sophistication caused the "mini" computer to gain in popularity as either a general purpose business machine or dedicated to a specific system. Among the most popular of the "mini" computers were:

- IBM's System 36/38 series (which led to the AS/400)
- DEC PDP Series (which gave way to the DEC VAX/VMS)
- Hewlett-Packard's HP-3000 series with MPE
- Data General Eclipse series with AOS
- PRIME

The competition was fierce in the "mini" market which resulted in considerable product improvements and better value to the customer. Instrumental to the success of the mini was the adoption of UNIX as developed by Bell Labs, a powerful multi-user, multitasking operating system that eventually was adopted by most, if not all, mini manufacturers.

But the major development in computer hardware was not the mainframe, nor the mini; it was the "micro" computer which was first popularized by Apple in the late 1970's. IBM countered with the its Personal Computer (PC) in the early 1980's. At first, the micro was considered nothing more than a curiosity but it quickly gained in popularity due to its inexpensive cost, and a variety of "apps" for word processing, spreadsheets, graphics, and desktop publishing. This caught on like wildfire as micros spread through corporate desktops like the plague.

By the mid-1980's the "micro" (most notably the PC) had gained in power and sophistication. So much so, that a series of graphical based products were used for software development in support of the Structured Programming movement of the 1970's. Such tools were dubbed "CASE" (Computer Aided Software Engineering) which allowed developers to draw their favorite software diagramming technique without pencil and paper. Early CASE pioneers included Index Technology, Knowledgeware, Visible Systems, Texas Instruments, and Nastec, as well as many others. CASE tools took the industry by storm with just about every MIS organization purchasing a copy either for experimental use or for full application development. As popular as the tools were initially, there is little evidence they produced any major systems but, instead, helped in the design of a single program.

Recognizing the potential of the various CASE tools, IBM in the late 1980's devised an integrated development environment that included IBM's products as well as third parties, and entitled it "AD/Cycle." However, IBM quickly ran into problems with the third party vendors in terms of agreeing on technical standards that would enable an integrated environment. Consequently, the product ran aground not long after it was launched. In fact, the prosperity of the CASE market was short-lived as customers

failed to realize the savings and productivity benefits as touted by the vendors. By the early 1990's, the CASE market was in sharp decline.

Instead, companies turned to Programmer Workbenches which included an all-in-one set of basic tools for programming, such as editing, testing, and debugging. Microsoft and Micro Focus did particularly well in offering such products.

Data Base Management Systems also took a noticeable turn in the 1980's with the advent of "relational" products involving tables and keys. The concept of the "relational" model was originally developed by IBM Fellow and mathematician Edgar (Ted) Codd in a paper from 1970. The concept of a relational DBMS was superior to the earlier network and hierarchical models in terms of ease of use. The problem resided in the amount of computer horse-

(continued on page 6)

(continued from page 5)

power needed to make it work, a problem that was overcome by the 1980's. As a result. new DBMS products such as Oracle and Ingres were introduced which quickly overtook their older competitors. There was an initial effort to convert DBMS mainstays such as TOTAL, ADABAS, and IDMS into relational products, but it was too little, too late. As for IBM, they simply re-labeled their flagship product, IMS, as a "transaction processor" and introduced a totally new offering, DB2, which quickly dominated the DBMS mainframe market.

Program generators continued to do well during the 1980's but it was during this period that 4GL's (fourth generation languages) were introduced to expedite programming. The 4GL was a natural extension of the DBMS and provided a convenient means to develop programs to interpret data in the data base.

Another development worth noting is the evolution of the Data Dictionary into "Repositories" (also referred to as "Encyclopedias") used to store the descriptions of all of an organization's information resources. One of the motivating factors behind this was IBM (for AD/Cycle) who realized they needed some sort of cohesive bond for the various CASE tools to interface. This is another area pioneered by MBA who introduced their "PRIDE"-Enterprise Engineering Methodology (EEM) to study a business and formulate an Enterprise Information Strategy, and their "PRIDE"-Data Base Engineering Methodology (DBEM) to develop the corporate data base, both logically and physically. To implement these new methodologies, their "PRIDE"-LOGIK Dictionary was expanded to include business models, and data models. By doing so, MBA renamed "PRIDE"-LOGIK the "PRIDE"-IRM (Information Resource Manager) which complemented their concept of Information Resource Management.

In terms of the MIS infrastructure, two noteworthy changes occurred; first was the introduction of the Chief Information Officer (CIO) as first described in the popular book, *"Information Systems Management In Practice"* (McNurlin, Sprague) in January 1986. Basically, the MIS Director is elevated to a higher management level where, theoretically, he/she is operating on the same level as the Chief Operating Officer (COO), and Chief Financial Officer (CFO) for a company. In reality, this has never truly happened and, in many cases, the title "CIO" is nothing more than a change in name, not in stature. The second change is the change in job title of "Programmer" to "Software Engineer." Again, we are primarily talking about semantics. True, many of the programmers of the 1980's studied Structured Programming, but very few truly understood the nature of engineering as it applies to software, most are just glorified coders. Nonetheless, the "Software Engineer" title is still actively used today. In contrast, the last of the true "Systems Analysts" slowly disappeared. Here too is evidence of the change of focus from systems to software.

During the 1980's we also saw the emergence of MBA's graduating from the business schools and working their way into the corporate landscape. Although they didn't have an immediate impact on the systems world, they had a dramatic effect on the corporate psyche. Their work resulted in severe corporate cutbacks, downsizing, and outsourcing. This changed the corporate mindset to think short-term as opposed to long-term. Following this, companies shied away from major systems projects (such as the MIS projects of the 1960's) and were content tackling smaller programmer assignments, thus the term "app" was coined to describe a single program application.

Interestingly, a "quality" movement flourished in the 1980's based on the works of W. Edwards Deming and Joseph M. Juran who pioneered quality control principles in the early part of the 20th century. Unfortunately, their early work was unappreciated in America and, consequently, they applied their talents to help rebuild the industrial complex of postwar Japan. It was only late in their lives did they receive the recognition of their work in the United States (after Japan became an economic powerhouse). Another influential factor was the introduction of the ISO 9000 standard for quality management which was originally devised by the British and later adopted as an international standard. Little attention would probably have been paid to ISO 9000 if it weren't for the fact that European businesses started to demand compliance in order to conduct business with their companies.

Nevertheless, these factors resulted in a reorientation of American businesses to think in terms of developing quality products which, inevitably, affected how systems and software were produced. The real impact of the quality movement though wouldn't be felt in the systems world until the next decade.

To summarize the 1980's from a systems development perspective, the focus shifted away from major systems to smaller programming assignments which were implemented using newly devised CASE tools. This fostered a "tool-oriented approach" to development whereby companies spent considerably on the latest programming tools but little on management and upfront systems work. In other words, they bought into the vendor's claims of improved programmer productivity through the use of tools.

(continued pn page 7)

(continued from page 6)

Unfortunately, it resulted in patchwork systems that required more time in maintenance as opposed to modifying or improving systems. "Fire fighting" thereby became the normal mode of operation in development.

1990's - REDISCOVERY

As the PC gained in stature, networking became very important to companies so that workers could collaborate and communicate on a common level. Local Area Networks (LAN) and Wide Area Networks (WAN) seemed to spring-up overnight. As the PC's power and capacity grew, it became obvious that companies no longer needed the burden of mainframes and minis. Instead, dedicated machines were developed to control and share computer files, hence the birth of "client/server computing" where client computers on a network interacted with file servers. This did not completely negate the need for mainframes and minis (which were also used as file servers), but it did have a noticeable impact on sales. Companies still needed mainframes to process voluminous transactions and extensive number-crunching, but the trend was to move away from big iron.

Thanks to the small size of the PC, companies no longer required a big room to maintain the computer. Instead, computers were kept in closets and under desks. This became so pervasive that companies no longer knew where their computer rooms were anymore. In a way, the spread of computers and networks closely resembled the nervous system of the human body.

One of the key elements that made this all possible was the introduction of Intel's 30386 (or "386") chip which allowed 32-bit processing. To effectively use this new technology, new operating systems had to be introduced, the first being IBM's OS/2 in the late 1980's. OS/2 provided such things as virtual memory, multitasking and multithreading, network connectivity, crash-protection, a new High Performance File System, and a slick object oriented desktop. Frankly, there was nothing else out there that could match it. Unfortunately, Microsoft bullied its way past OS/2 with Windows 95 & NT. By the end of the 1990's, OS/2 was all but forgotten by its vendor, IBM. Nevertheless, it was the advent of 32-bit computing that truly made client/server computing a reality.

Another major milestone during this decade was the adoption of the Internet by corporate America. The Internet actually began in the late 1960's under the Department of Defense and was later opened to other government and academic bodies. But it wasn't until the 1990's that companies started to appreciate the Internet as a communications and marketing medium.

The first web browser was developed by Tim Berners-Lee in 1990 which led to the World Wide Web protocol on the Internet. Early web browsers included Mosaic, Netscape Navigator, and Microsoft's Internet Explorer, among others. The beauty of the Internet was that all computers could now access the Internet regardless of the operating system, making it a truly universal approach to accessing data. To write a web page, a simple tag language was devised, Hyper Text Markup Language (HTML), which was compiled at time of request to display the web page. HTML was nice for developing simple static web pages (not much interaction, just simply view the web page). Developers then invented new techniques to make a web page more dynamic thereby allowing people to input data and interact with files, which ultimately allowed for the merchandising of products over the Internet.

Wanting to do something more sophisticated through the web browser, Sun Microsystems developed the Java programming language in 1995. Java was a universal programming language that could run under any operating system. Their mantra was *"Write once, run anywhere."* This was a radical departure from programming in the past where it was necessary to recompile programs to suit the peculiarities of a particular operating system. Basically, Java made the operating system irrelevant, much to Microsoft's chagrin. Further, Java could be used in small pocket devices as well as in the new generation of computers powering automobiles. This did not sit well with Microsoft who ultimately fought the propagation of Java.

By the 1990's the Structured Programming movement had fizzled out. Instead, "Object Oriented Programming" (OOP) gained in popularity. The concept of OOP was to develop bundles of code to model real-world entities such as customers, products, and transactions. OOP had a profound effect on Java as well as the C++ programming language.

During this time, source code generators faded from view. True, companies were still using report writers and 4GL's, but the emphasis turned to "Visual Programming" which were programming workbenches with screen painting tools to layout inputs and outputs.

The Relational DBMS movement was still in high gear, but the use of Repositories and Data Dictionaries dropped off noticeably. Of interest though was the introduction of "Object Oriented Data Base Management System" (OODBMS) technology. Like OOP, data was organized

(continued on page 8)

(continued from page 7)

in a DBMS according to real-world entities. Regardless, Relational DBMS dominated the field.

Also during this decade "Data Mining" became popular whereby companies were provided tools to harvest data from their DBMS. This effort was basically an admission that companies should learn to live with data redundancy and not be concerned with developing a managed data base environment.

Because of the radical changes in computer hardware and software, companies became concerned with their aging "legacy" systems as developed over the last thirty years. To migrate to this new technology, a movement was created called "Business Process Re-engineering" (BPR). This was encouraging in the sense that companies were starting to think again in terms of overall business systems as opposed to just programs. I'm not sure I agree with the use of the term "Re-engineering" though; this assumes that something was engineered in the first place (which was hardly the case in these older systems).

Nonetheless, CASE-like tools were introduced to define business processes. Suddenly, companies were talking about such things as "work flows," "ergonomics," and "flowcharts," topics that had not been discussed for twenty years during the frenzy of the Structured Programming movement. Ultimately, this all led to the rediscovery of systems analysis; that there was more to systems than just software. But by this time, all of the older corporate Systems Analysts had either retired or been put out to pasture, leaving a void in systems knowledge. Consequently, the industry started to relearn the systems theory, with a lot of missteps along the way.

Companies at this time were still struggling with devising a suitable development environment. Most were content with just maintaining their current systems in anticipation of the pending Y2K (Year 2000) problem (where date fields were to change from 19XX to 20XX which could potentially shutdown companies). However, a few companies began to consider how to apply more scientific principles to the production of systems. Since people were already talking about "Software Engineering," why not apply engineering/manufacturing principles to the development of total systems?

Back in the early 1980's, Japan's Ministry of International Trade & Industry (MITI) coordinated a handful of Japanese computer manufacturers in establishing a special environment for producing system software, such as operating systems and compilers. This effort came to be known as Japanese "Software Factories" which captured the imagination of the industry. Although the experiment ended with mixed results, they discovered organization and discipline could dramatically improve productivity.

Why the experiment? Primarily because the Japanese recognized there are fundamentally two approaches to manufacturing anything: "one at a time" or mass production. Both are consistent approaches that can produce a high quality product. The difference resides in the fact that mass production offers increased volume at lower costs. In addition, workers can be easily trained and put into production. On the other hand, the "one at a time" approach is slower and usually has higher costs. It requires workers to be intimate with all aspects of the product.

MBA took it a step further by introducing their concept of an "Information Factory" in the early 1990's. The Information Factory was a comprehensive development environment which implemented MBA's concept of Information Resource Management. Basically, they drew an analogy between developing systems to an engineering/manufacturing facility, complete with assembly lines, materials management and production control. These concepts were proven effective in companies throughout Japan, most notably Japan's BEST project, which was sponsored by the Ministry of Finance. As background, the ministry wanted to leapfrog the west in terms of banking systems. To do so, they assembled a team of over 200 analysts and programmers from four of the top trust banks in Japan; Yasuda Trust & Banking, Mitsubishi Trust & Banking, Nippon Trust & Banking, and Chuo Trust & Banking. By implementing MBA's concepts they were able to deliver over 70 major integrated systems in less than three years. Further, because they had control over their information resources using a materials management philosophy, the Y2K problem never surfaced.

In terms of infrastructure, development organizations essentially went unchanged with a CIO at the top of the pyramid and supported by Software Engineers and DBA's. But there was one slight difference, instead of being called an MIS or IS department, the organization was now referred to as "IT" (Information Technology). Here again, the name hints at the direction most organizations were taking.

Finally, the 1990's marked a change in the physical appearance of the work force. Formal suit and ties gave way to casual Polo shirts and Docker pants. At first, casual attire was only allowed on certain days (such as Fridays), but it eventually became the normal mode of dress. Unfortunately, many people abused the privilege and

(continued on page 9)

(continued from page 8)

dressed slovenly for work. This had a subtle but noticeable effect on work habits, including how we build systems.

2000's - GADGETS

We are now past the halfway point in this decade and there is nothing of substance to report in terms of computer hardware, other than our machines have gotten faster, smaller, with even more capacity. Perhaps the biggest innovation in this regard is the wide variety of "gadgets" that have been introduced, all of which interface with the PC, including: Personal Digital Assistants (PDA's), iPods, MP3 players, digital cameras, portable CD/DVD players (and burners), cell phones, PS2 and XBox game players. These devices are aimed at either communications or entertainment, giving us greater mobility, yet making us a bit dysfunctional socially. All of this means the computer has become an integral part of our lives, not just at work but at home as well.

Shortly after taking the reigns of IBM in 2003, CEO Sam Palmisano introduced "On-Demand Computing" as the company's thrust for the years ahead and, inevitably, it will mark his legacy. The concept as described by Palmisano was simple, treat computing like a public utility whereby a company can draw upon IBM for computing resources as required. "On-Demand Computing" made a nice catch-phrase and was quickly picked up by the press, but many people were at a loss as to what it was all about. Some of the early developments resulting from IBM's "e-Business On Demand" research included balancing the load on file servers, which makes sense. But IBM is carrying the analogy perhaps too far by stressing that "on demand" is the manner by which companies should run in the future. Basically, the theory suggests we abandon capacity planning and rely on outside vendors to save the day. Further, it implies computers supersede the business systems they are suppose to serve. Instead of understanding the systems which runs a business, just throw as much computer resources as you need to solve a problem. This is like putting the cart before the horse.

The "on-demand" movement has evolved into "Service Oriented Architectures" (SOA) where vendors are introducing "on-demand" applications that will take care of such tasks as payroll, marketing, etc. through the Internet. Again, it all sounds nice, but as far as I can see, this is essentially no different than service bureaus like ADP who for years provided such processing facilities. Now, companies are being asked to swap out their internal programs for third party products. I fail to see how this is different than buying any other packaged solution, other than an outsider will be taking care of your software.

The need to build software faster has reached a feverish pitch. So much so, full-bodied development methodologies have been abandoned in favor of what is called "Agile" or "Extreme Programming" which are basically quick and dirty methods for writing software using power programming tools. To their credit, those touting such approaches recognize this is limited to software (not total systems) and is not a substitute for a comprehensive methodology. Agile/Extreme Programming is gaining considerable attention in the press.

Next, we come to "Enterprise Architecture" which is derived from a paper written by IBM's John A. Zachman who observed that it was possible to apply architectural principles to the development of systems. This is closely related to consultants who extoll the virtues of capturing "business rules" which is essentially a refinement of the Entity Relationship (ER) Diagramming techniques popularized a decade earlier using CASE tools.

As in the 1990's, concepts such as "Enterprise Architecture" and "business rules" is indicative of the industry trying to reinvent systems theory.

CONCLUSIONS

Like computer hardware, the trend over the last fifty years in systems development is to think smaller. Developers operate in a mad frenzy to write programs within a 90 day time frame. Interestingly, they all know that their corporate systems are large, yet they are content to attack them one program at a time. Further, there seems to be little concern that their work be compatible with others and that systems integration is someone else's problem. Often you hear the excuse, *"We don't have time to do things right."* Translation: "*We have plenty of time to do things wrong."* Any shortcut to get through a project is rationalized and any new tool promising improved productivity is purchased. When companies attempt to tackle large systems (which is becoming rare) it is usually met with disaster. Consequently, companies are less confident in their abilities and shy away from large system development projects.

Corporate management is naive in terms of comprehending the value of information and have not learned how to use it for competitive advantage (unlike their foreign competitors). Further, they are oblivious to the problems in systems development. They believe their systems are being developed with a high degree of craftsmanship, that they are integrated, and that they are easy to main-

(continued on page 10)

(continued from page 9)

tain and update. Executives are shocked when they discover this is not the case.

The problems with systems today are no different than fifty years ago:

- End-user information requirements are not satisfied.
- Systems lack documentation, making maintenance and upgrades difficult.
- Systems lack integration.
- Data redundancy plaques corporate data bases.
- Projects are rarely delivered on time and within budget.
- Quality suffers.
- Development personnel are constantly fighting fires.
- The backlog of improvements never seems to diminish, but rather increases.

Although the computer provides mechanical leverage for implementing systems, it has also fostered a tool-oriented approach to systems development. Instead of standing back and looking at our systems from an engineering/manufacturing perspective, it is seemingly easier and less painful to purchase a tool to solve a problem. This is like taking a pill when surgery is really required. What is needed is less tools and more management. If we built bridges the same way we build systems in this country, this would be a nation run by ferryboats.

The impact of the computer was so great on the systems industry that it elevated the stature of programmers and forced systems people to near extinction. Fortunately, the industry has discovered that there is more to systems than just programming and, as a result, is in the process of rediscovering basic systems theory. Some of the ideas being put forth are truly imaginative, others are nothing more than extensions of programming theory, and others are just plain humbug. In other words, the systems world is still going through growing pains much like an adolescent who questions things and learns to experiment.

I have been very fortunate to see a lot of this history first hand. I have observed changes not just in terms of systems and computers, but also how the trade press has evolved and the profession in general. It has been an interesting ride.

Throughout all of this, there have been some very intelligent people who have impacted the industry, there have also been quite a few charlatans, but there has only been a handful of true geniuses, one of which was Robert W. Beamer who passed away just a couple of years ago. Bob was the father of ASCII code, without which we wouldn't have the computers of today, the Internet, the billions of dollars owned by Bill Gates, or this document.

END

About the Author

Tim Bryce is the Managing Director of M. Bryce & Associates (MBA) of Palm Harbor, Florida and has 30 years of experience in the field of Information Resource Management (IRM). He is available for training and consulting on an international basis.

"PRIDE" Special Subject Bulletins can be found at:

http://www.phmainstreet.com/mba/mbass.htm

They are also available through the "PRIDE Methodologies for IRM Discussion Group" at:

http://groups.yahoo.com/group/mbapride/

You are welcome to join this group if you are so inclined.

The "Management Visions" Internet audio broadcast is available at:

http://www.phmainstreet.com/mba/mv.htm

Also, be sure to read Tim's Blog at:

http://blogs.ittoolbox.com/pm/irm/