Fourth
Semantic Interoperability
for
E-Government Conference

February 9 & 10, 2006

Semantics in Metadata Repository
and
Systems Integration Efforts

David Eddy & Associates, Inc.
P.O. Box 57320
Babson Park, MA 02457-0320
781-455-0949

deddy@davideddy.com

Page intentionally left blank.

Semantics in Metadata Repository and Systems Integration Efforts

BACKGROUND[1]

In the December 1993 issue of *American Programmer*, Ed Yourdon reported his impressions from a visit to American Subsidiary, Inc. in southern California. He touched on their remarkable long-term focus, corporate culture, and unusually successful application of enterprise metadata repository technology. This article addresses ASI's use of a central metadata repository to support system development efforts and shows how sound metadata management practices give an organization increased systems flexibility to respond to ever changing market conditions.

AMERICAN SUBSIDIARY, INC. (ASI)

ASI is a wholly owned subsidiary of Global Conglomerate, Ltd., (GCL) of Japan. At $8.8 billion in 1992 sales, GCL is listed as #166 among the Global Fortune 500. ASI's 1992 sales of $507 million place it just below the U.S. Fortune 500, which begins at $585 million. ASI employs 400 people, with 50 in MIS, who are split evenly between support and maintenance/development.

ASI is the North American distribution arm for GCL. Their business is primarily logistics—moving finished goods through the distribution pipeline to end consumers.

ENTERPRISE ARCHITECTURE

In 1977 Rose Twohey (executive vice president) joined ASI and quickly hired Tom Antel (MIS director). Twohey describes the development/maintenance process at ASI at the time as "out of control." Her charter was to (1) restore stable service levels and (2) implement an effective software planning and development[2] process.

Twohey quickly brought in **John Zachman** (of Zachman Information Architecture[3] fame) from IBM to do a Business Systems Plan (BSP). Twohey and Antel had attended an IBM presentation that spoke of "managing data as a fundamental corporate resource." For reasons now lost in the mists of time, the vision of developing and maintaining "data as a valuable corporate resource" has stayed firmly in their sights ever since. The journey, however, has not been without some potholes.

[1] Originally published in the March, 1994, Vol. 7 no 3 issue of Ed Yourdon's American Programmer. Names have been changed in this 2006 version.

[2] A wise programmer once observed, "Development is everything that comes before the first compile of the first program in the first system. After that, you're into maintenance."

[3] Known as Enterprise Architecture (EA) in the early 21st century.

The year-long BSP drill entailed taking a high-level view of the organization's process and data requirements with objectives and constraints superimposed. When done well, the matrices resulting from a BSP expose top management to the otherwise invisible complexity and interrelatedness of their systems.

While working on the BSP, ASI also purchased a metadata repository with two specific objectives. First, the metadata repository would support the database administration function for their then new IMS database applications. Second, it would be used to automate the process of defining its data element inventory. This second effort was driven by the requirement to communicate concisely and on an ongoing basis the semantics (informational content and meaning) of the data being sent to their Japanese parent.

Fired by the revelations of the BSP experience, but not yet understanding the BSP was not an end in itself, a concerted attempt was made to implement the BSP matrices in the repository. While technically correct and robust, this two year effort proved to have no lasting value because it was unwieldy, and, from the perspective of Twohey's MIS department, did not add value to the primary business of satisfying user needs on a day-to-day basis. Antel, as senior manager most directly involved in both the BSP effort and the ultimate solution, describes the top-down versus bottom-up conundrum thusly: "We, as managers, are inherently top-down thinkers. We conceptualize beautiful solutions from 100,000 feet. We feel we're drowning in details if we push all the way down to 80,000 feet. However, what sticks in the long run is the implementation of nitty-gritty details that directly support the daily routines of the troops in the trenches."

WHAT WORKED—THE REPOSITORY AS CHANGE-LOG[4]

Use of the repository required more than buying a capable tool. Many other companies have purchased a repository, expended lavish resources on it, and produced no lasting benefit for the enterprise. What did ASI do right? **Automating their existing manual change-log process is what worked for ASI**. It is an inescapable fact in all software efforts there is a development life-cycle[5]—whether or not the organization chooses to formally recognize it. ASI chose to weave their change-log process into the repository to such an extent that today programs cannot be put into production without going through a series of clearly defined—and always obeyed—steps that are directly controlled from the repository and the surrounding automated change control procedures.

Although ASI's change-log does not sound as grand as a "life-cycle methodology," in fact what they have done is implement an automated life-cycle methodology in their repository. ASI software libraries have three formal states: test/development, acceptance, and production. Promotion between libraries is entirely controlled by the automated change-log/repository process. When a project is initiated, it is given an ID number in the repository. All subsequent

[4] Change-Log meaning software configuration management (SCM).

[5] Also Systems development life-cycle (SDLC).

work is tracked to this project number. The components (copybooks[6], programs, job-steps, datasets, etc.) impacted by the project are tracked to the project number. Prior to initiating a project, the extent of the project's impact is researched in the repository. With the repository now containing more than two decades of "dynamic artifacts" about virtually all previous development projects, ASI has found that researching prior projects is a reliable method to both avoid reinventing the wheel and to have a solid understanding how big a project really is.

There are three primary control steps in ASI's automated change-log process: First, all data elements are formally defined in the repository. Second, all copybooks are defined in the repository and can contain only properly defined data elements.[7] Third, all programs and their components (i.e. copybooks, subroutine calls, file/database assignments, etc.) are registered in the repository. Without going into the mind numbing technical details, a critical process control step happens when a development project is approved and is promoted from test to acceptance. When the program is recompiled from the acceptance libraries, copybook definitions must now come directly from the repository, not the programmer's private test library. If a programmer has used undefined copybooks or unofficial data elements, the program fails to compile. A programmer has to experience only once the embarrassment of a supposedly working program not compiling to be convinced that the rules must be obeyed.

This promotion process is easy-to-describe and hard-to-implement with airtight automation. Only management can oversee putting such procedures into place. Only management can ensure that these procedures are always followed. Project managers, programmers, and end users all eventually want to bypass such control steps "just this once for an emergency fix." If management capitulates to the inevitable political pressure—"I've got Mr. Big's authorization signature to bypass standard procedures!"—to ignore these checkpoints, the repository inexorably becomes inaccurate and not worth using as a reliable corporate memory and impact analysis tool.

REPOSITORY INFRASTRUCTURE

For the promotion process to happen seamlessly, a great deal of attention was given to automating the repository update process. It took several years to get the repository into self-sustaining mode.[8] The primary technical architect and implementer, John Shipley, states that "...anytime within the first four years the whole effort could have fizzled and totally disappeared." His key to success was being able to weave into the change-log process a series of scanning programs that automatically keep the metadata repository 100% synchronized with production systems. Shipley realistically recognized programmers would (1) not do additional

[6] The COBOL term "copybook" is synonymous with "include" for those who speak C.

[7] A key semantic step had to be separately automated since repositories do not have such functionality.

[8] One of the success hurdles is when the original technical sponsor moves on to other duties. Historically most repository efforts fail at this point.

repository documentation work requested of them and (2) what work they did do would be of questionable quality. Shipley's approach was to require as little additional manual intervention as possible and to automate as many of the documentation steps as technically feasible.

Shipley, as data administrator, and Richard Herder, as database administrator, were uncompromising in their objective of assuring maximum accuracy via automated scanning. Antel, as MIS Director, provided unfailing backup because he understood what was at stake. They all recognized that unless the scanners worked successfully behind the scenes, the repository, with its ability to do ad hoc impact analysis queries, would be seen as just more make-work documentation to be trivialized and eventually ignored by programmers.

When the repository implementation began in the late 1970's ASI already had an existing legacy portfolio and work force in which some programmers used copybooks and some did not. Rather than futilely mandating "Thou shalt henceforth use copybooks!" to veteran programmers, Shipley's approach was to control and document programs where existing copybooks were already used. Over time, as resources permitted, in-line data structures were converted to copybooks on a project-by-project basis. There was no massive, frontal assault to do the job all at once. **Eventually, both project managers and programmers came to recognize that when copybooks and data elements were more completely documented in the repository, their daily routine was easier because they now had access to accurate and complete analysis information.**

The impact of these efforts is best seen by comparing industry norms with ASI results. Dr. Howard Rubin in his "Black Hole"[9] metrics study, discovered that at fewer than 20% of the 2,000 sites he studied can management articulate the scope/extent of their software portfolios. In sharp contrast, Herder can state authoritatively that ASI's systems portfolio consists of approximately 20 applications; 5,300 programs; 3,200 records; 16,000 datasets; and 5,600 data elements.[10] Additionally, **ASI has precise definitions of what their data elements mean**.[11] Consequently, ASI is able to connect data according to its core business meanings, despite the fact that the data elements have many different technical names and representations across the functional applications. The benefit of this semantic capability is that ASI knows in detail the components in its systems, how they are interrelated, and what they mean.

[9] "Inside the Information Systems Black Hole", The Rubin Review Volume V Issue 3, 4th Quarter 1992.

[10] One result of this tight control was the very lost cost for Y2K repairs at 40 cents/LoC.

[11] As a COBOL shop, they use the PRIME-MODIFIER-CLASS scheme from IBM's 'OF Language.' Twelve required CLASS words are: CODE, NUMBER, AMOUNT, TEXT, CONSTANT, CONTROL, COUNT, DATE, FLAG, NAME, PERCENT and TITLE. The complete vocabulary—automatically enforced by a software process—was approximately 1,600 terms in 2005.

BENEFITS

ASI insists they have been unable to measure the financial value the central change-log/metadata repository infrastructure produces. Although there are no documented financial measures, the indirect results are clearly visible. The best indicator of the value of ASI's efforts unfolded as this article was being written.

In early December 1993 the parent GCL decided as of January 1994 ASI would be responsible for distributing a line of hydraulic engines in the U.S. Without the "dynamic artifacts" project record in the repository, ASI would have been faced with three increasingly less attractive and more expensive choices: (1) take a wild guess and inform GCL the systems expansion effort would take at least six months, praying their guesstimate was reasonably accurate, (2) support the new product manually, or (3) outsource the distribution effort.

Because the repository contains an accurate blueprint of system components, ASI was able to look at a previous product-line expansion project. Being able to examine a detailed log of the impact and extent of a previous project enabled ASI to state with confidence they would be able to extend the existing core systems to support the new product line in less than one month. The ability to respond in a timely manner to a significant addition to their business is considered benefit enough by ASI management.

An additional indirect benefit of ASI's careful maintenance of its core systems is the systems are integrated, not cloned. In other organizations a classic mode of coping with unexpected business opportunities and demands and short deadlines is to clone (copy en mass) a core system that is "pretty close" to the new requirements and then apply radical modifications. After doing this a few times, organizations find themselves stuck with a series of systems or pieces of systems that appear to have a common ancestry but now work in subtle to radically different ways. Although this classic quick-and-dirty approach does get a "new" system in place rapidly, the longer term maintenance costs and increasing lack of flexibility become significant burdens.

A further problem with the clone approach, which ASI has consciously avoided, is over time technical personal become increasingly dedicated to a narrow sliver of functional knowledge. They know their corner of a system but cannot be moved easily to similar applications.

CONCLUSION

ASI's success with their repository chronicles how one organization put into place the structure to facilitate, and indeed require, good systems development practices. Their achievement has not come overnight—they have been working on this solution for 36 years. They have not had the luxury of lavish resources. While it is certain the dedication and long-term focus of the individual participants contributed in no small way to their success, the principles they followed—the basics of sound life-cycle management—are universally applicable to all organizations that must build and maintain software systems, large and small.

ACKNOWLEDGEMENTS

Many thanks to the now disguised individuals, plus John Zachman for their valuable time in helping me research this article.

BIOGRAPHY

David Eddy & Associates, Inc., specializes in missionary sales and marketing efforts for niche information resource management products. Mr. Eddy has 15 years of software development experience in banking, insurance, and consulting environments. From 1989 until 1993 he was dedicated to sales and marketing of repository-based products. His efforts enabled a client to expand its customer base beyond North America into Japan and Europe. He actively maintains contact with Fortune 1000 firms and personally manages a database of 7,000 contacts at 2,500 sites, focusing on professionals in reverse engineering and information resource management.

Mr. Eddy has an M.B.A. from Babson College and a B.A. in Russian history from Union College.

With the collapse of the first generation metadata repository tools market in the early 1990s, in October 1994 Mr. Eddy focused on Year 2000 issues. His missionary marketing efforts for Year 2000 awareness were described in the August 25, 1995 issue of the Wall Street Journal in Tom Petzinger’s Front Lines section. He has testified to the Senate Committee on Small Business.