DoD Legacy Systems: Reverse Engineering Data Requirements

Peter Aiken† Alice Muntz* & Russ Richards†

† Defense Information Systems Agency
Center for Information Management
1951 Kidwell Drive, Fifth Floor
Vienna, VA 22182
phaiken@vcu.edu

* Hughes Information Technology Company
Information Systems Technology Laboratory
PO Box 92919
Los Angeles, CA 90009
ahmuntz@hac2arpa.hac.com

Outline

Abstract

As with most large organizations, the Department of Defense has both strategic and economic needs to capitalize on and to consolidate existing information systems. This paper reports on a framework currently being used to reverse engineer selected legacy information systems in DoD's heterogeneous environment. This generic approach was developed to recover organizational business rules, business domain information, system functional requirements, functional dependencies, and system data architectures, largely in the form of normalized logical data models. We are applying the approach as a series of pilot studies on systems ranging from those using home grown languages and database management systems developed during the late 1960's to those using high order languages and commercial network database management systems. These pilot studies are being used to validate and refine the framework with real-life systems; to develop a baseline approach for reverse engineering DoD legacy information systems; and to scope and estimate future system re-engineering costs. Furthermore, the results of these projects will help to determine the economic viability of re-engineering, reverse, and forward engineering efforts for these legacy systems.

Keywords

Design, reverse engineering, software and system requirements and specifications, data architecture, business rules, data modeling

Introduction

The Department of Defense currently maintains more than 1.4 billion lines of code associated with thousands of heterogeneous, non-combat information systems at more than 1,700 data centers [10, 11]. This enormous inventory of often functionally duplicative systems creates two key classes of problems:

- The cost of operating these systems consumes an enormous portion of total DoD information technology spending--currently more than $9 billion annually.
- Despite this level of spending, the Department is often unable to obtain correct information from the data stored in the various existing databases due to a lack of standardized data and data structures across systems. Submitting the same query to each of the more than 20 payroll systems can result in not just multiple answers *but in multiple kinds of answers*. At times, consolidating the query responses has proven to be an impossible task.

Over the years this base of information systems has been continuously modified to implement changing needs including: functional requirements, business rules, and data architectures. While few currently satisfy DoD-wide needs, some of these systems satisfy many department-wide requirements and most contain invaluable information. The Center for Information Management/Data Administration Program Management Office (CIM/DAPMO) recognized the value of these systems and identified the need to use reverse engineering analysis and technologies to recover and reuse functional requirements, and potentially system components, to develop an information base for data migration planning. CIM/DAPMO initiated a project to develop a reverse engineering framework (consisting of procedures, methods, and a tool set) and to validate the framework in a series of reverse engineering projects using prominent DoD legacy information systems. The project objectives were to:

1. develop a systematic framework for reverse engineering database and data structures to recover business rules, domain information, functional requirements, and to use these data products to develop normalized logical data models, construct data architectures, and document data requirements;
2. validate and refine the framework in a DoD specific context by reverse engineering a number of DoD legacy systems;
3. develop a means of scoping and estimating future data reverse engineering project costs; and
4. develop and capture useful metrics to assist decision makers in determining the economic feasibility of future reengineering, reverse, and forward engineering efforts as DoD begins the process of consolidating and modernizing its inventory of information systems.

This paper describes the reverse engineering data requirements framework and experience applying it to five large scale DoD legacy information systems, each containing several million lines of code, and thousands of data structures. Section two describes the impact of the legacy environment complexities on the data reverse engineering process. Section three describes the reverse engineering project context. Section four describes the data reverse engineering framework, and approaches to modeling and model management. The last section relates some lessons learned from experience with the framework and the contributions that reverse engineering can make to the applications development life cycle.

Impact of Legacy Environment Complexity

Operational Complexity

Historically, DoD has allowed world-wide subordinate level organizations to maintain separate, functionally duplicative systems supporting operational requirements for various DoD components including: the Army, Navy, Air Force, Marine Corps, Joint Chiefs of Staff, Unified and Specified Commands, Agencies etc. To collect, analyze or process data at higher organizational levels, management collected it from lower levels. The collection process depends on subordinate organizations feeding information upward--often manually. The format of the upward feed must be meticulously specified at each level. Outside of limited specific mission critical instances, the consistency and accuracy of data flows has been practically impossible to maintain and control. The impact on reverse engineering projects include the following:

1. The inventory of physical evidence is difficult to collect and analyze. Even if documentation exists, it is often outdated or of poor quality. In addition, personnel with the required knowledge are often no longer available.
2. The existing interface data elements defined for transferring data instances do not represent completely identified, sharable data structure and semantic requirements among systems. In order to obtain a complete picture of the data sharing requirements, reverse engineering analysis must determine not only how the interface data elements were generated and used among systems but also identify other non-interface data elements which are synonymous among systems and are therefore sharable.
3. Even though a specific system may have been selected to replace a group of legacy systems in a functional domain, the implementation will not be successful unless the legacy system unique critical requirements are identified, documented and addressed by the replacement system.

Technological Complexity

The current DoD legacy systems inventory includes obsolete electronics, technology, and systems designs--up to 30 years old and poorly documented. Many of these decades-old designs are pushing the limits of their engineered capabilities creating reliability and maintainability problems. Designation as migration systems capable of serving the entire DoD has added functional and technical requirements. Characteristics of the five systems (reverse engineered in phase 1 of this project) are shown in Figure 1. Moreover, they cannot be readily adapted to open architectures and current technologies.

For instance, one of the systems uses application program managed memory overlays. This originally was an innovative use of flat-file technology, using a table-driven approach to separate process from data as database technology has done. However, a fixed record length limits the number of fields available to accommodate the growing user requirements. Currently this system is using the same field for multiple uses with different meanings depending on the user groups. Allowing individual user discretion rather than enterprise standards, permitted a restrictive physical data limit to serve more customers, but this approach is reaching its design and operation limit.

System	Language(s)	Data Handling System(s)	Operational Environment
Defense Civilian Personnel Data System (DCPDS)	Burroughs assembly and home-grown proc-edural language: "Samuel"	Home-grown database management system	Multiple sites using remote access
Defense Civilian Pay System (DCPS)	COBOL	IDMS/R	
Marine Corps Total Force System (MCTFS)	COBOL & Assembly Language Code	VSAM & Adabase	
Composite Health Care System (CHCS)	MUMPS	FILEMAN	
Medical Performance Factors (MEPRS)	MUMPS	FILEMAN	

Figure 1: Example Legacy Environment.

Administrative Complexity

Richard Nolan [8] stated, "In the stages of control and integration, the dominant forces have to do with organizational discipline and don't relate very closely to technology." The Department of Defense is no exception. The coordination, negotiation, management approval, buy-in processes for reverse engineering are exceptionally complicated and time consuming but critical to the success of reverse engineering projects. This is due to the existence of numerous system stake holders (e.g., planners, owners, builders, maintainers, existing and future users.) Figure 2 illustrates the variety and number of people who required coordination and briefings for a single reverse engineering project. In this example, we were forced to coordinate with points of contact from 11 different organizations. We used these "Client Mazes" to track communication for the projects. The impact administrative complexity on reverse engineering projects includes:

CUSTOMER	CONTRACTOR ISSUES REPRESENTATIVES	PERFORMANCE TEAM	FUNCTIONAL PROPONENT	SYSTEM IMPLEMENTERS
Office of the Director of Defense Information Deputy Director DD Reps.	**DoD Contract Holder** Contract Officer (CO) Contracting Officer Rep (COR) Contracting Officer Tech Rep (COTR) **ISSAA** Govt. Legal rep **Reverse Engineered System Contractor's Office/Legal Dept.** Contractor's VP Legal Dept. **Contractor Contract Office/Legal Contracting Office** Legal Advisor	**DISA/CIM/ DAPMO** Project Manager Lead Engineer Functional Liaison **HITC** Program Manager Project Area Leader Lead Engineer	**Program Office of the reverse engineered system** PM System POC Database POC Deputy PM **DoD Functional Area** Data Administrator Planning Division Strategic Planner	**Contractor holding Development Contract for reverse engineered system** Program Manager Product Dev Lead Engineer 5 Members of Technical Staff **Federally Funded Research Center** Lead Engineer Group Leader 2 Engineers

Figure 2: Representative Client Maze.

1. The strategic objectives of the system with respect to each system stake holder must be identified and prioritized. The reverse engineering objectives and priorities must be synchronized with and agreed to by the stake holders.
2. Without coordinated "buy-in," a reverse engineering project cannot be successful. High level management approval is necessary but not sufficient. Mid-level management and systems personnel must also understand and support the reverse engineering project objectives or else "rice-bowl syndromes" may jeopardize the project's success.
3. The negotiation, planning, and buy-in processes must be done "before" the project starts.

Project Context

Figure 3 depicts DoD's approach [3,4] to eliminate redundant systems by having functional steering committees selecting "migration" systems from the existing information system inventory. Migration systems should implement the majority of functional and data requirements of a class of legacy systems having the same, similar, or overlapping information and/or domain. The migration systems eventually become functional area "target" systems by:

- adding the remaining essential functional requirements for other legacy systems--before phasing them out,
- separating data from process, and
- incorporating DoD standard structures for data reuse and data sharing.

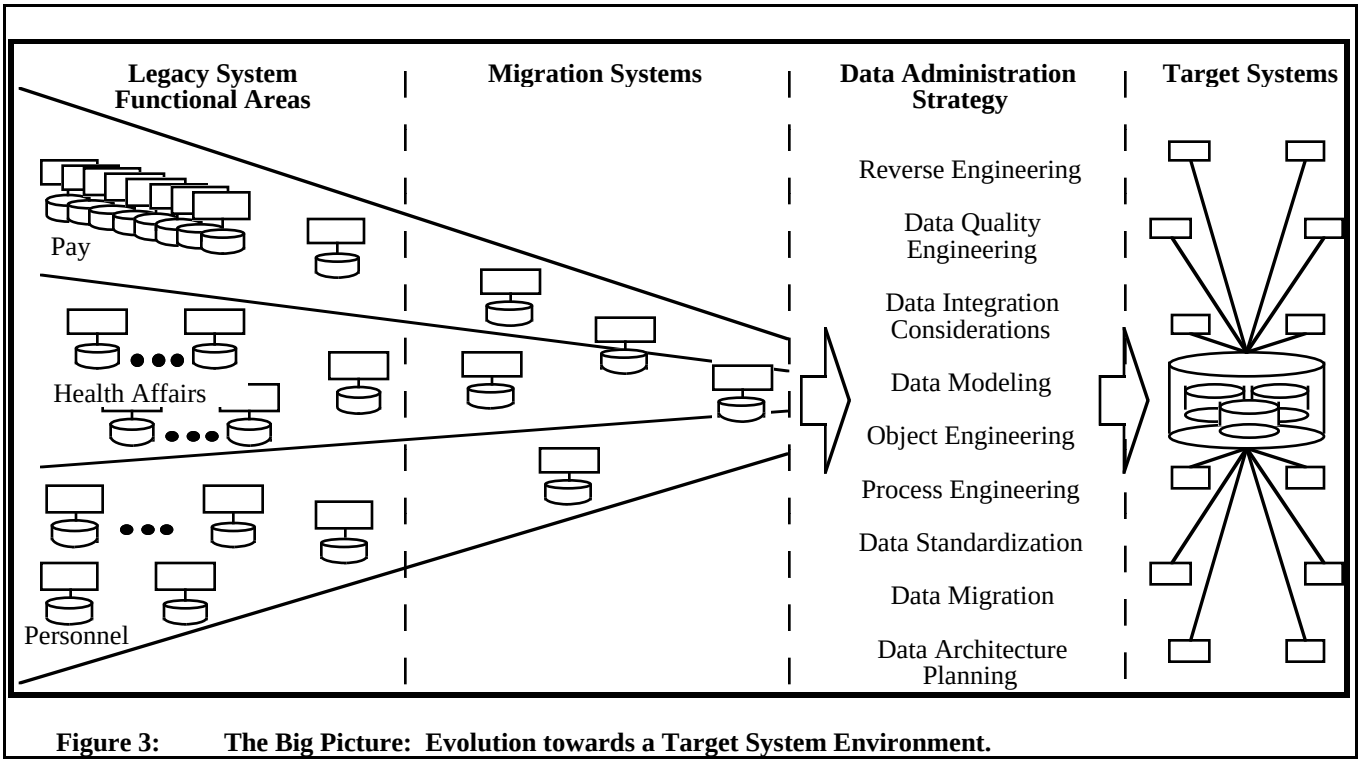


Figure 3: The Big Picture: Evolution towards a Target System Environment.

Most of the legacy systems were developed without the process models or data models--now needed to support data standardization. DoD now requires that target systems use logical data models to represent data requirements. Process and data models must be developed to represent

the policies, strategies, and tactics of organizational operation. Under the framework development of the models include identification, refinement, validation, and linking of all business functions, policies, rules, and activities to data elements, model content, and physical evidence. This approach ensures that all data structures can be identified and linked to supported processes and minimizes data impact when processes change. Reverse engineering in this framework results in "engineered data" and directly supports DoD data migration and data standardization efforts.

Using the ANSI three-schema architecture paradigm, Figure 4 shows the relationship of the reverse engineered logical view (logical) to the physical and user (external) views of the target environment. Reverse engineering is the approach selected to derive logical data models from migration systems; even though this approach is generally used to optimize applications code design and streamline system operations. Using this approach the reverse engineering framework, also identifies, extracts, and integrates the unique critical requirements contained in non-designated legacy systems into the appropriate system migration path. As shown in Figure 5, these critical requirements (indicated by scattered dots in the legacy data "environment") will be incorporated into the migration system data requirements as a part of the data migration efforts.

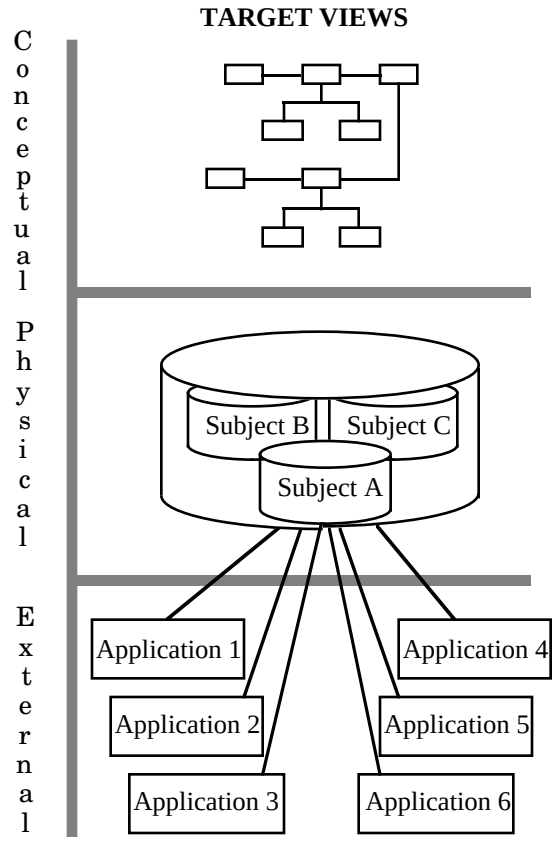


Figure 4: Three Schema View of Target Environment.

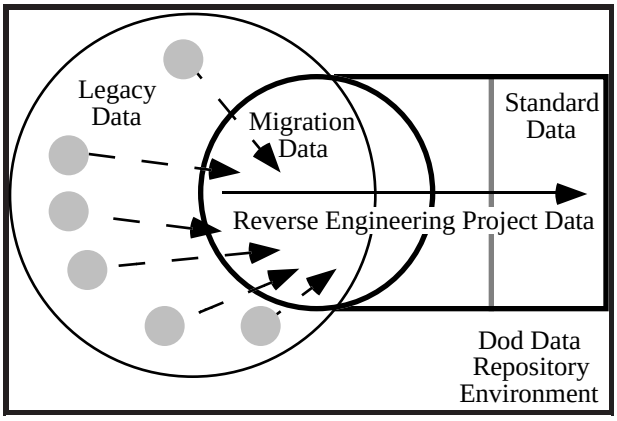


Figure 5: Systems and Data Migration to Support Data Standardization.

To evolve a designated migration system to a target system, a migration system must be forward engineered using many reverse engineering products, additional requirements identified from the business reengineering activities and must also support standard open systems architectures.

Basing the forward engineering on the data models developed from the reverse engineering analysis will make the migration system more adaptable and easier to maintain. Adopting an open systems approach should reduce maintenance costs and improve system reliability and maintainability.

Figure 6 depicts the various activities completed in the reverse engineering projects reported here. For each functional area a set of information systems were originally designed to satisfy component specific operational requirements rather than DoD-wide strategic requirements. Reverse engineering processes recovered "as-is" requirements. Business reengineering activities conducted by functional area working groups have been and will continue defining "to-be" global business process and logical data models, i.e., the operational requirements of "to-be" DoD-wide information systems. Selected migration system data assets will be migrated (some enhanced) into these "to-be" systems.

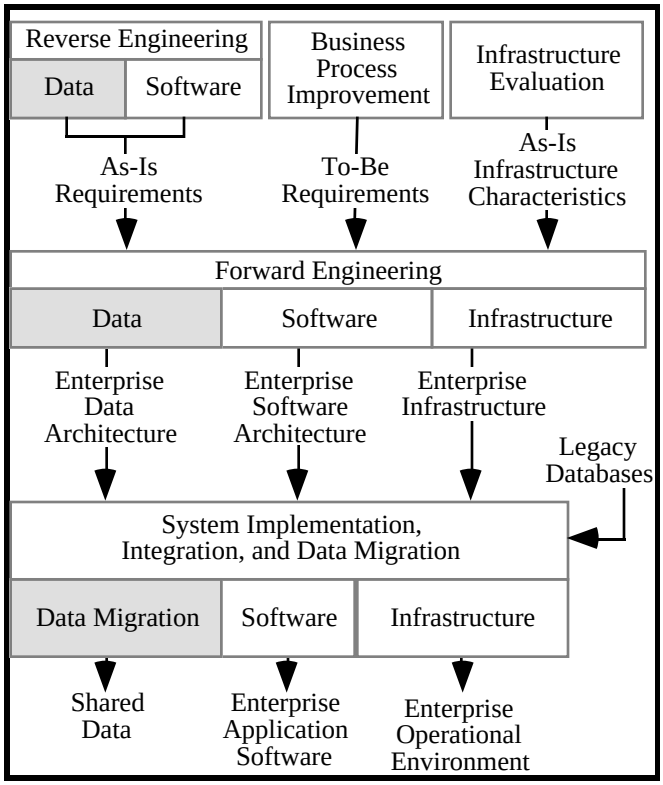


Figure 6: Scope of Reverse Engineering Efforts.

Currently, there is no direct mapping between data elements and organizational business rules, business domain information, system functional requirements, functional dependencies, and organizational data distribution architectures. "As-is" data elements and their embedded business requirements are often in conflict with or are insufficient to satisfy the "to-be" business requirements. Reverse engineering the "as-is" requirements of the migration systems is essential to recovering the associated business requirements at the operational, tactical, and strategic levels.

Recovered "as-is" requirements are then compared against the "to-be" business requirements to identify business requirement gaps during forward engineering. In addition to identifying

business requirement gaps, technological gaps are also crucial to determining if "as-is" migration systems satisfy system operational performance requirements and maintenance cost constraints. The migration system architectures are evaluated against technical requirements to identify technical requirement gaps. As part of forward engineering, the business requirements gaps, technical requirements gaps, and data element quality are evaluated to determine the "migratability" and "integrability" of migration systems forming the basis of economic justification to forward engineer specific systems.

The reverse engineering program supports the integrated analysis and redesign/development activities required to modernize the selected migration systems. Due to the massive and complex nature of the DoD systems inventory, modernization must be conducted in multiple phases. The project validated and refined the reverse engineering framework using five selected DoD information systems in three functional areas: Personnel, Pay, and Health Affairs. The overall thrust of the effort was to identify cross-functional "human being" related data elements within DoD for integration purposes.

As shown in Figure 7, the Defense Civilian Personnel Data System (DCPDS) is in Personnel, the Defense Civilian Payroll System (DCPS) is in Pay, the Marine Corps Total Force System (MCTFS) is cross functional between Personnel and Pay. The Composite Health Care System (CHCS), Coordinated Care Performance (CCP), and Medical Expense and Performance Reporting System (MEPRS) are all in Health Affairs.

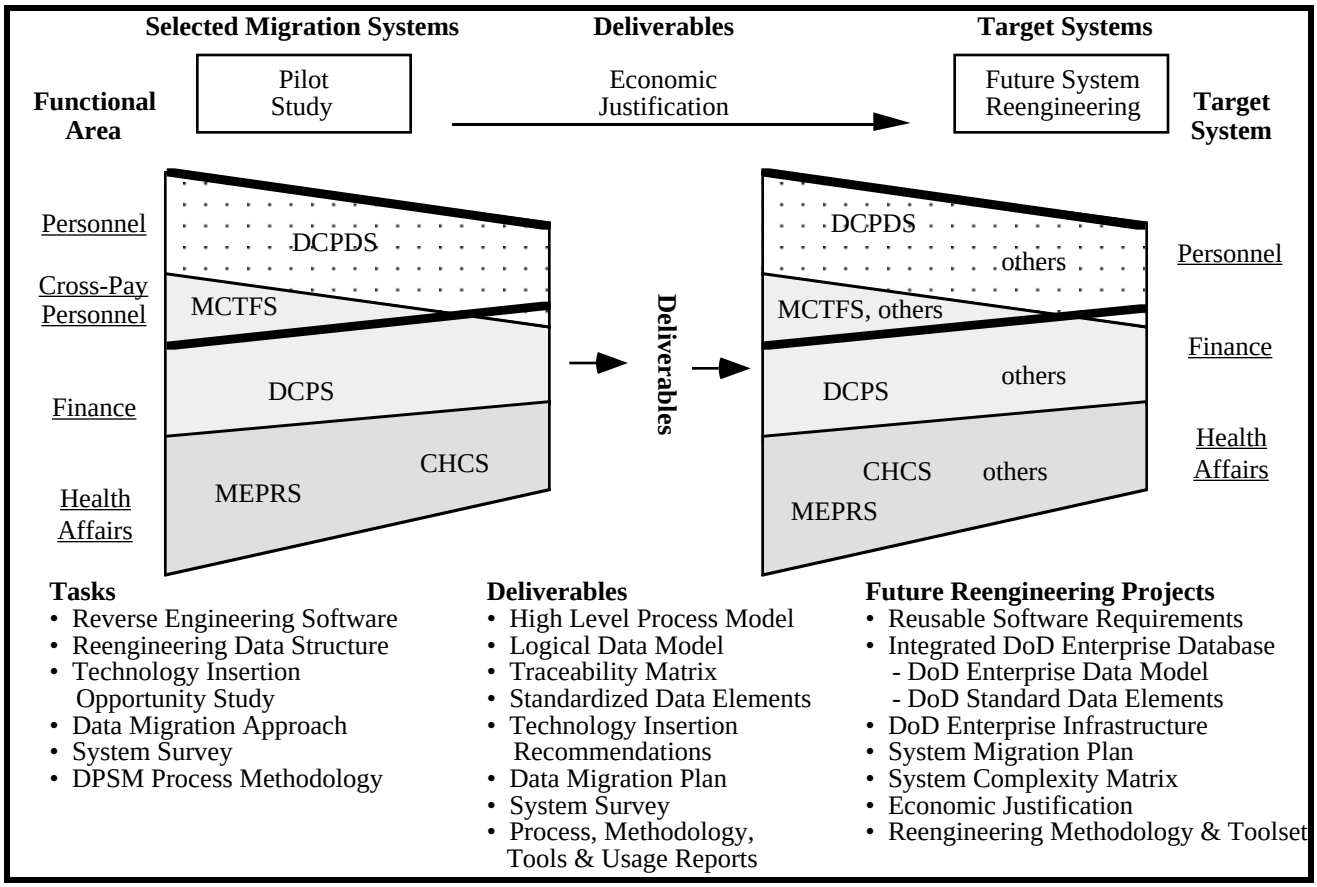


Figure 7: Migration Systems Context.

Before starting the project, quantitative measures of system size and complexity were unknown. This was one of the reasons for running this project as a prototype effort before beginning full-scale reengineering efforts. During the initial phase of the project we developed a cost model for reverse engineering data requirements. We also recorded performance metrics and system complexity measures for each system. We used the metrics to continuously tune-up our cost model. The tuned-up cost models can now used to estimate a full-scale data reengineering effort. The metric-based results become input to economic analyses to determine selection and economic evaluation criteria for future projects in other functional areas.

Project focus was on derivation of normalized, logical data models and standardization of data element names for generic elements (attributes) and prime elements (entities). The logical data models to support standardization of data elements were used to represent organizational business rules, business domain information, system functional requirements, functional dependencies, and system data architectures of the selected system. We also performed a limited cross-functional integration for a selected business domain (calculate pay) relevant to civilian personnel and civilian pay functional areas. In addition to the cross-system integration, specific objectives were:

- requirements analyses, reengineering analysis and design, rapid prototyping and testing of systems and subsystems, and consideration of data migration and integration issues;
- an extendible inventory of migration system data assets;
- a realistic and extendible approach for reverse engineering the remainder of DoD migration systems;
- identification of automated tool requirements for use in other reverse engineering projects.

The project also supported the extension of the DoD data model which will feed the DoD Data Repository, in a format consistent with DoD Data Standardization procedures [4].

Integration Considerations

One of the challenges for cross-functional integration for a selected business function (e.g., calculate pay) is to identify what data elements are relevant to integration. We discovered that interface data elements are not the only relevant data elements for an integrated data model for personnel and pay systems. Figure 8 illustrates our approach to isolate the "calculate pay" data elements in order to construct a logical data model for shared information between civilian pay (DCPS) and civilian personnel (DCPDS) systems.

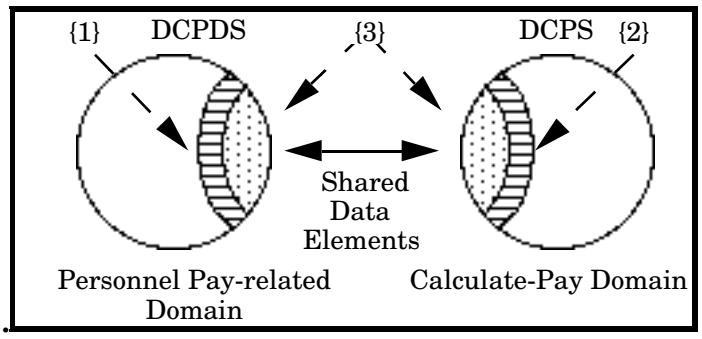


Figure 8: Domain Relationship for Cross System Integration for Two Systems.

Since the decision of what to pay a person is represented in personnel policy and how to pay is a pay function, we first started with DCPDS to identify {1} the system functions and data elements relevant to paying a person. These data elements were collectively called the "personnel pay-related domain" and included: staffing, affirmative employment, job classification, and employee management relations. The second step {2} was to identify the system functions and relevant data elements for calculating pay within DCPS. The third step {3} was to compare the resultant data elements from the first step {1} and second step {2} to identify *shared data elements*, i.e., the overlapping elements between DCPDS and DCPS for the "calculate pay" business function. For cross functional integration among three or more systems they were compared in pair-wise fashion iteratively using the same technique as illustrated in Figure 8 to identify shared data elements. In multiple systems comparisons, more extensive analysis is required because business domains, business rules, functional requirements, and functional dependencies must be cross compared among systems at each stage to isolate not "shared" but "identical" data elements and structures.

Reverse Engineering Data Requirements Framework

As stated previously, the complexity of the legacy environment has great impact on the feasibility and success of a reverse engineering project. Management of these complexities is shown in the first four steps of the framework process depicted in Figure 9. Steps 5-10 outline the process for discovering data requirements and recovering normalized data models from the collected physical evidence. The recovered logical data model associates business rules, business domain information, system functional requirements, functional dependencies, and organizational data distribution architectures and data elements.

One of the essential outcomes of reverse engineering is a traceability matrix linking the data model components to the physical evidence supporting their existence. The traceability matrix is critical for validating the correctness of derived logical models from physical evidence. Later the traceability matrix can be used to identify the impacts of process change and to plan data migration from legacy to migration to target systems. Using CASE tools, the logical data models can be used to automatically create table structures for the selected DBMS while the traceability matrix is used to develop the mapping required to migrate data from the legacy systems to the appropriate tables in the new system.

Based on the characteristics of the systems we encountered, the key aspects of the technical approach included: (1) divide-and-conquer; (2) extraction of business rules from software and data structures; (3) model management; (4) configuration management; and (5) schema integration. Although software was divided into software modules used to implement various functional requirements, business functions were often implemented as part of several different software functions.

As part of the physical implementation, data structures also have multiple roles. Data structures directly extracted from software were often transient data variables used to store processing results from stored data elements. Processing would be an implementation of business rules or low level mathematical algorithms (e.g., sort). Data structures are also used to hold data presented as on-line screens or reports and can be used to map to data elements stored in databases or files. Hence, data structures defined in software and data dictionaries, if they exist, may represent only some of the conceptual, external, or physical views of data elements.

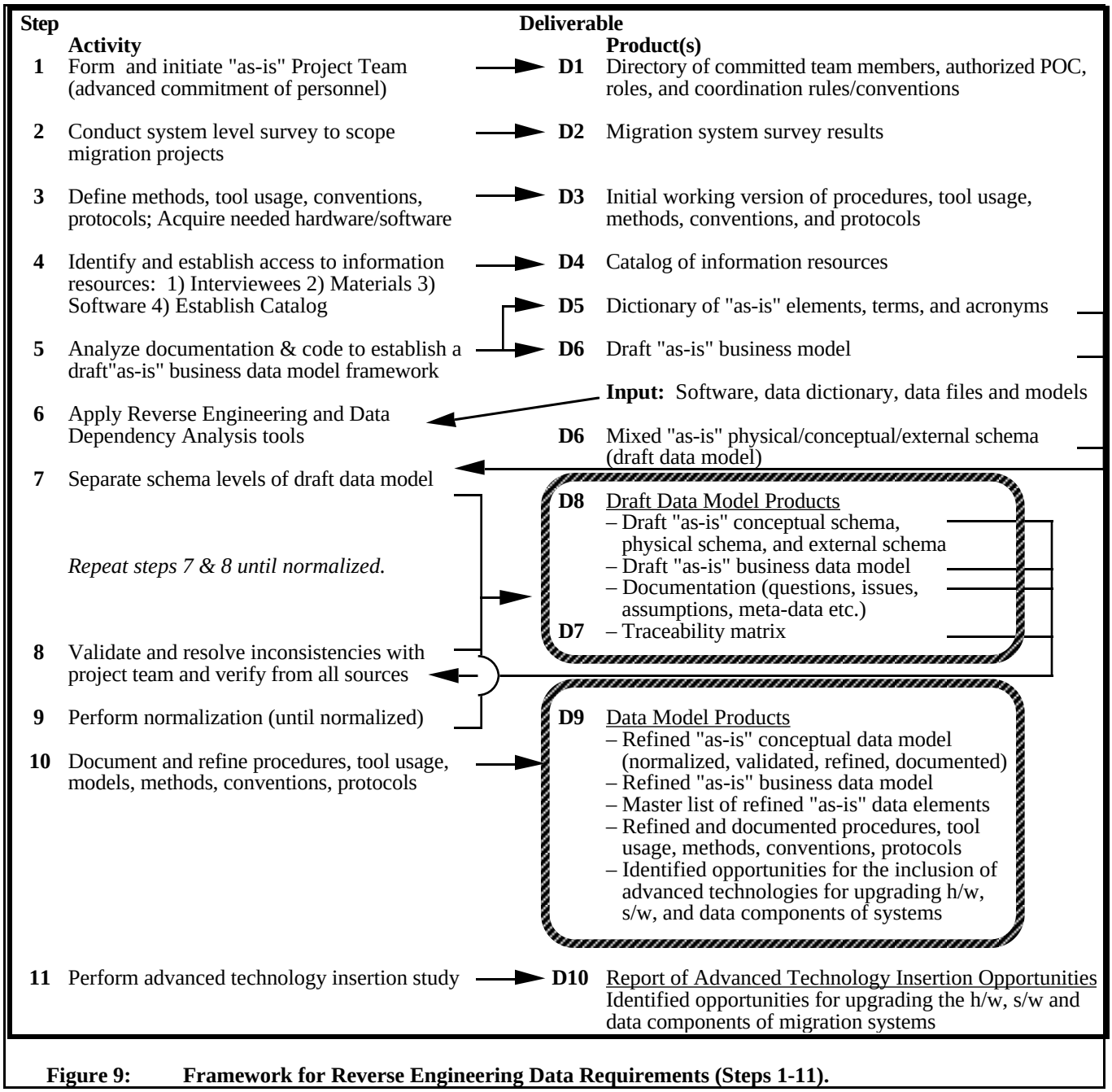


Figure 9: Framework for Reverse Engineering Data Requirements (Steps 1-11).

The most difficult aspect of reverse engineering is to discover business rules and data entities from software and data structures. Massive quantities of data structures and the associated code, force a divide-and-conquer approach to discover data elements and organize them into categories. Our approach is top-down then bottom-up. During the top-down step, we analyzed material relevant to the conceptual view (e.g., user screens, reports, policy statements). This helped to establish draft high level "as-is" business process and data model frameworks. These were used to quickly identify a set of conceptual buckets for "holding" relevant categories of data structures for a specific domain. We used data buckets (entities) derived from the business process model to partition the business data model into views.

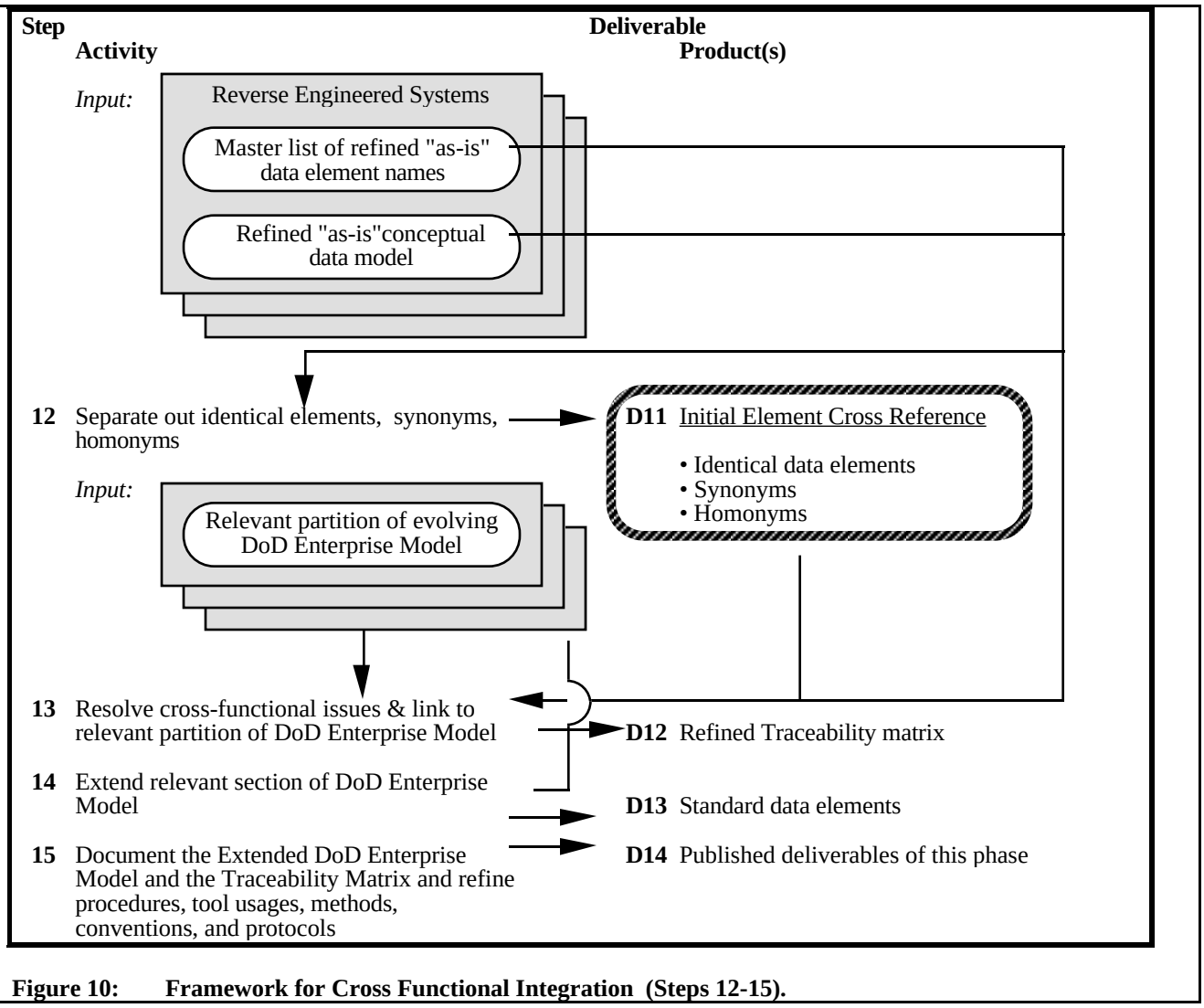


Figure 10: Framework for Cross Functional Integration (Steps 12-15).

Each data model view corresponds to a business process. The functional dependencies between views are inherited from the relationships between processes in the high level process model. We also identified the non-transient data structures access (create, read, update) aspects of processes. We then derived more detailed logical data models and linked these data structures to the derived data entities using the traceability matrix. All data structures and data entities were linked to their associated operational, tactical, and/or strategic requirements. For transient data structures computed from non-transient data elements and later used for updating other non-transient data, we defined structure entities to capture the business rules associating these non-transient data elements.

Reverse engineering analyses must also carefully examine and analyze a system after technology insertion or improvement ideas are surfaced. Step 11 in Figure 9 identifies this opportunity. This step should not be overlooked. When the objective for reverse engineering also includes cross-functional integration, steps 12-15 in Figure 10 are to be used for integrating the data models for the selected systems.

As shown in Figure 11, each version of a reverse-engineered data model is associated with an encyclopedia. Our model management approach defined standardized policies and procedures

making it feasible for reverse engineering team members to review each other's work and understand information in other project encyclopedia/directory structures. The encyclopedias store information in three separate dictionaries. The Plan Dictionary contains planning information, the Data Dictionary contains the logical data models and associated information, and the Design Dictionary contains physical structures and related information.

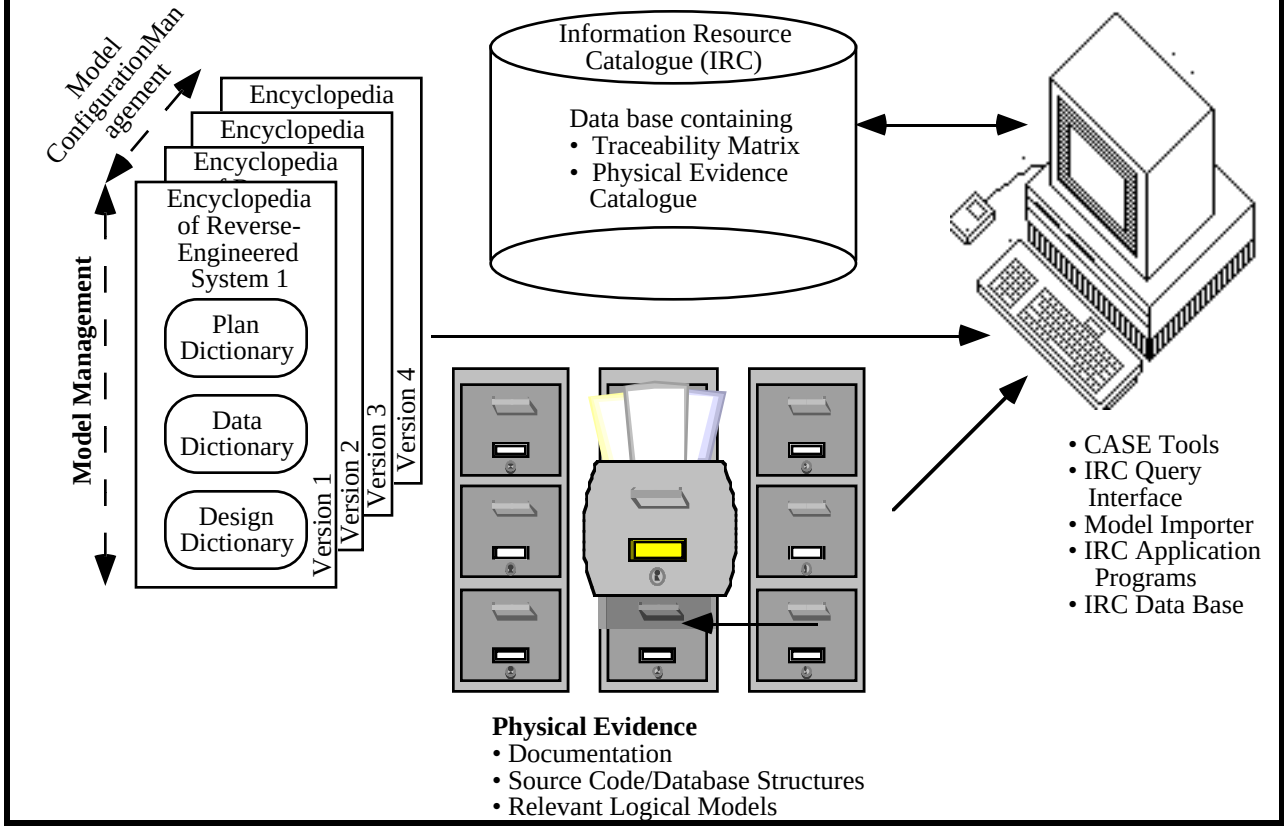


Figure 11: Reverse Engineering Modeling Approach and Model Management.

To enhance traceability, physical evidence obtained is linked in an Information Resource Catalogue (IRC) database. The IRC contains sources of information relevant to each reverse-engineered system including system manuals, source code, directives, interview results, etc. It provides an electronic index for the information resources gathered during the reverse-engineering life cycle. The information resources are physically stored in filing cabinets. The traceability matrix is used to identify and/or trace the correlation of items contained in the various models and document the satisfaction of business requirements and rules. The data models developed using the IE: Advantage CASE tool may be imported to the IRC. The traceability matrices stored in the encyclopedia are also loaded into the IRC. The IRC also permits users to link physical evidence to the data model with an interface that users can use to query the contents of IRC, the data modeling status, and linkage between physical elements, logical entities, and business requirements and rules.

Lessons Learned

Management Issues

Getting formal commitment and authorization from the system stakeholders (especially administrative and technical management) is a crucial factor to the success of reverse engineering projects. One primary problem was associated with lack of access to key system personnel because they were simultaneously required in forward development efforts. In addition, the costs to properly reverse engineer systems and value of the reverse engineering products are consistently under estimated by management. A common misperception is that we just "run the code through a CASE tool and the tool will produce new systems." It has been challenging to convince management that reverse engineering is a substantially broader and more complex task than just "restructuring the code" [l, 2]. Restructuring the code is useful to improve the system maintainability, but it does not facilitate data migration, evolution of a component based (e.g., Air Force, Army) system to satisfy DoD-wide system requirements, data sharing among functional areas (e.g., Health Affairs, Personnel, Pay), and incorporation of new or changed data requirements resulting from business process improvement activities.

The current crop of CASE tools touted as reverse engineering solutions, focus on associated physical data structure and variables to segments of code. This is useful in identifying how physical schema are created, updated, read, and deleted, but it is not sufficient to construct the logical and conceptual external views of data requirements. Even if the functional and technical experts help the reverse engineers clean up the analysis products, the products do not include, for instance, the links to physical evidence provided in the traceability matrix and model management support described here as services offered by the framework. More importantly, using such tools in isolation may be as damaging as performing inadequate, inaccurate, or incomplete systems or software requirements engineering. Recovering a normalized logical data model together with the associated business rules, policies, and physical data structures is hard, human-performed analysis even with this systematic reverse engineering framework (including model management and appropriate modeling approach). CASE tools may augment the analysis to provide initial understanding of physical implementation of the system, but CASE tools will not provide "the" solution for recovering the conceptual and logical data requirements.

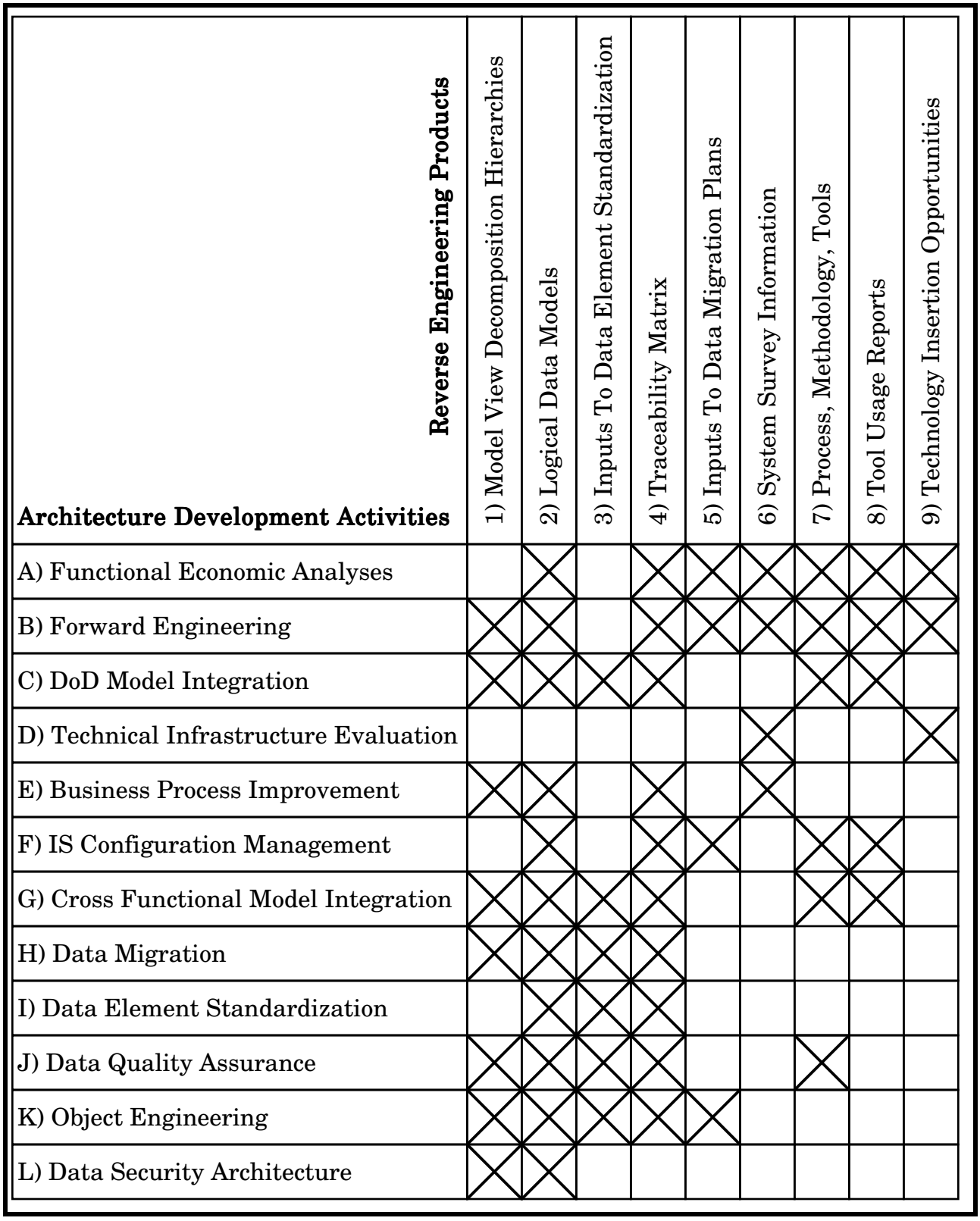
From our experience, currently no single CASE tools is capable of handling this diversified, and complex environment. In order to discover and recover data requirements implemented in these systems, major analysis has been performed by humans and supplemented with commercial and custom software in this approach.

Lack of Understanding of Reverse Engineering Outputs

Throughout the initial phase of this project, we have been continuously asked:

- What reverse engineering products are produced?
- How can these products be used?
- How is reverse engineering related to other system development activities?
- When will the reverse engineering products be ready?
- Why do they need reverse engineering in order to obtain standard data elements?

We constructed a matrix, Figure 12, to help answer some of these questions.



Architecture Development Activities / Reverse Engineering Products	1) Model View Decomposition Hierarchies	2) Logical Data Models	3) Inputs To Data Element Standardization	4) Traceability Matrix	5) Inputs To Data Migration Plans	6) System Survey Information	7) Process, Methodology, Tools	8) Tool Usage Reports	9) Technology Insertion Opportunities
A) Functional Economic Analyses		X		X	X	X	X	X	X
B) Forward Engineering	X	X		X	X	X	X	X	X
C) DoD Model Integration	X	X	X	X			X	X	
D) Technical Infrastructure Evaluation						X			X
E) Business Process Improvement	X	X		X		X			
F) IS Configuration Management		X		X	X		X	X	
G) Cross Functional Model Integration	X	X	X	X			X	X	
H) Data Migration	X	X	X	X					
I) Data Element Standardization		X	X	X					
J) Data Quality Assurance	X	X	X	X			X		
K) Object Engineering	X	X	X	X	X				
L) Data Security Architecture	X	X							

Figure 12: Reverse Engineering Products and their Uses.

Reverse Engineering Products

Logical data models and standard data elements are not the only reverse engineering products. The other outputs of the reverse engineering framework include:

- **High Level Model View Decomposition Hierarchies** - These may be used to size the system, scope the project, and define the logical data model views.
- **Traceability Matrix** - The matrix, maintained in the IRC (described previously), substantiates the requirement for specific data model components by linking them to specific authorizing statutes, regulations, policy guidance, etc. The matrix can be used to perform impact analysis when subsequent requirements changes occur.
- **Technology Insertion Recommendations** - As long as these well-qualified teams are analyzing legacy information systems, it seems entirely appropriate to take note of major areas where technology insertion recommendations would be useful components in the resulting migration plans. Areas such as advanced database and communications network technologies are typical recommendations.
- **Framework, Methodology, and Tool Usage Reports** - As stated earlier the language, data management (or handling) systems, implementation complexity, and strategic plan for each system are different. The generic framework is followed for reverse engineering all systems, but we identified the detailed approaches for deriving logical code and data models from structured analysis. During each phase we refined and revalidated the reverse engineering framework, methodology, and tool usage for each class of systems.
- **System and Data Migration Plans** - The systems migration plan prescribes the necessary steps to make the existing legacy system compliant with IE concepts based on guidance from existing directives [3, 4]. It is a plan for bridging the gap between the "as-is" legacy information system serving a narrow purpose and the "to-be" integrated Department-wide information system.

- **Reusable Software Requirements** - Another by-product will be reusable software requirements in the form of data models, IRC domain specific rule sets, and perhaps eventually, the software constructed to support the requirements.

- **DoD Enterprise Data Model, DoD Standard Data Elements, Integrated DoD Enterprise Database** - These are all goals of the CIM/DAPMO data standardization effort. All outputs from the reverse engineering projects are integrated with these Department-wide efforts.

Data Standardization Misconceptions

A dangerous misperception is that data standardization is simply identifying associated data elements and their naming consistency in each system. When implemented, this causes future unforeseen, disastrous outcomes which are difficult to identify and isolate. To correctly use information from data elements, the business rules, policies, and the functional dependencies among elements must be identified and represented in a data model for each system. Before standardization can be achieved, model integration is an essential step to identify and resolve synonyms and homonyms based on the rules, policies and functional dependencies represented in the models. Without the integration step, incorrect information continues to propagate throughout the enterprise.

Lack Of Cost Estimation Approach For Reverse Engineering

Costs of the re-engineering efforts are difficult to estimate. One vendor we know charges a flat $1.00/line of code in the system. We were unable to even estimate the lines of code in the medical portion of the project because of the unstructured nature of the MUMPS programming language code. One measure of prowess among MUMPS programmers is how complex a program can be written with a single line of code. Perhaps this accounts for the various estimates in the number of lines of MUMPS code in CHCS ranging between 1.3 million to 2.5 million depending on whom you ask.

Implications Relevant To The Extension Of The System Life Cycle Process Model

Today's global economy imposes new requirements on legacy systems. Even though existing systems do not meet current needs, failure statistics of new software systems development indicate that new systems may also not meet user requirements. Integration, modernization, restructuring, and/or augmentation of the existing information infrastructure are the current trend to solving enterprise information integration problems. Regardless of the type of development life cycle (e.g., waterfall, spiral, JAD, RAD), the reverse engineering data requirements framework can contribute significantly throughout the life cycle.

Figure 13 depicts how the framework contributes to the traditional ("waterfall") information systems development life cycle. In order to achieve enterprise information integration, a set of activities (such as process engineering, data modeling, reverse engineering) must be in concert with the information system design and development phases.

Tool Assessment Relative to I-CASE: Finally, we believe that the current focus of reverse engineering CASE tools and CASE tool development efforts are concentrating on code analysis. While it is true that most organizations will have a more homogeneous information systems base than the Department of Defense, it is also true that much of the remainder of the federal government will have configurations similar to the Department of Defense and will also be unable to prescribe a single CASE tool solution for reverse engineering. A further criticism of the reverse engineering CASE tools is the focus on code analysis with little or no assistance for processes such as extracting business rules. We anticipate our work will be able to define a rudimentary set of requirements for reverse engineering CASE tools operating in a heterogeneous environment.

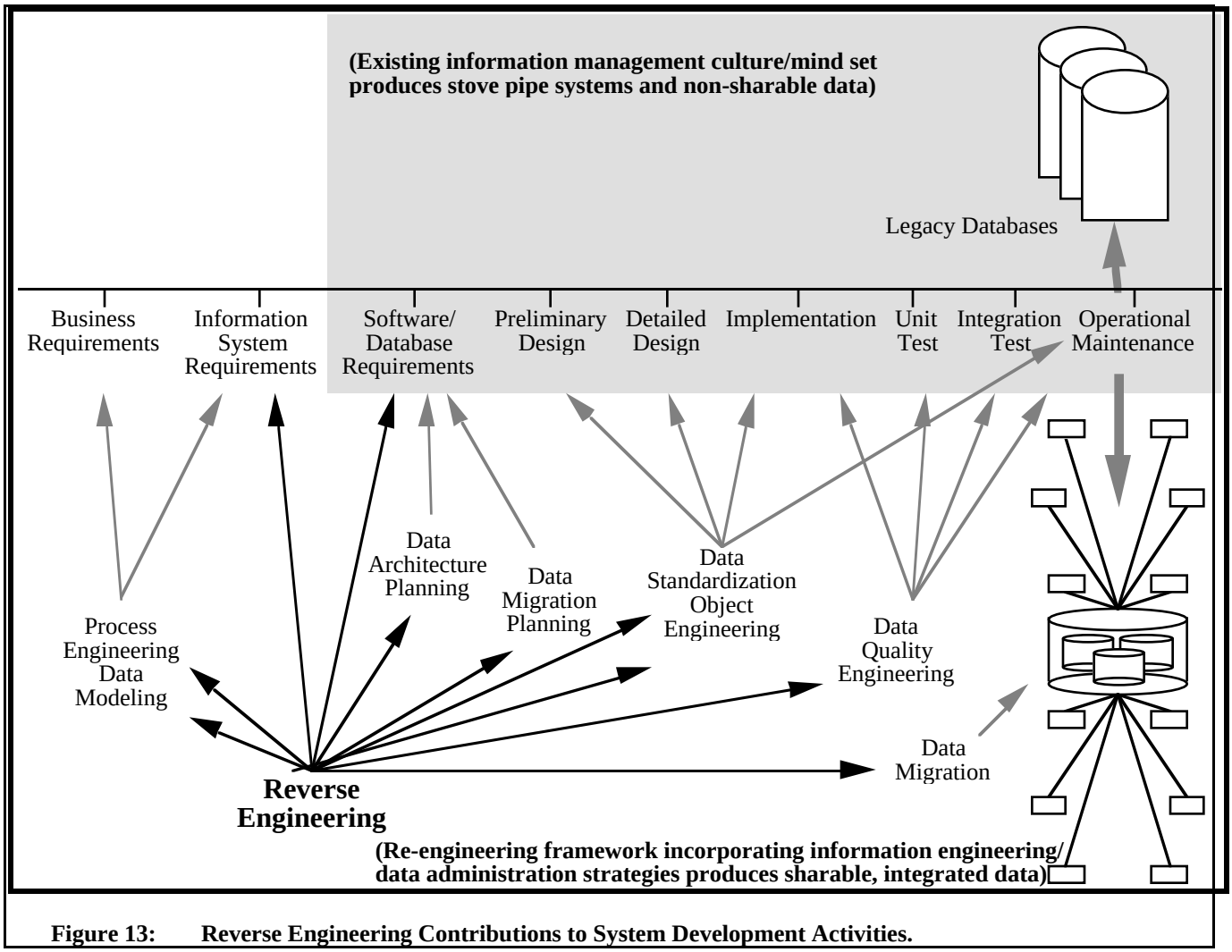


Figure 13: Reverse Engineering Contributions to System Development Activities.

Acknowledgments

The project team from Hughes Information Technology Company played a key role in developing and implementing the reverse engineering framework. The authors thank the dedication of: C. Haley, D. Heller, D. Hobson, K. Kawakami, K. Janes, S. Lem, B. Nguyen, T. Pace, C. Ramiller, C. Szymanski, C. Tholen. In addition, none of these reverse engineering projects could have achieved the real, tangible results they have without the cooperation of the respective DoD functional and technical communities and in particular: Civilian Personnel, Civilian Pay, Health Affairs, and Procurement. We acknowledge their participation in making Phase I of the reverse engineering program a success. Finally, the paper also benefited from comments and suggestions made by the Defense Finance and Accounting Data Administration Group.

References

[1] V. R. Basili and H. D. Mills "Understanding and Documenting Program" *IEEE Transactions on Software Engineering* May 1982. SE-8(3):270-283

[2] Elliot Chikofski and James H. Cross II "Reverse Engineering and Design Recovery: A Taxonomy" *IEEE Software* January 1990 7(1):13-17.

[3] DoD Directive 8020.1-M *Functional Process Improvement (for Implementing the Information Management Program of the Department of Defense*, August 1992 (Draft).

[4] DoD Directive 8320.1-M *Data Standardization*, August 1992 (Draft) .

[5] Richard E. Fairley *Software Engineering Concepts*, New York: McGraw-Hill Book Company, 1985.

[6] Clive Finkelstein *Information Engineering: Strategic Systems Development*, Addision Wesley Publishing Company, 1993.

[7] Donald C. Gause and Gerald M. Weinberg *Exploring Requirements: Quality Before Design*, Dorset House Publishing, New York, 1989.

[8] Richard Nolan "Managing the Crisis in Data Processing" *Harvard Business Review*, 1979.

[9] David Sharon "A Reverse Engineering and Re-Engineering Tool Classification Scheme" *Reverse Engineering Newsletter* Number 4 January 1993.

[10] Chris Staiti and Paul Pinella "DoD's Strassmann: The Politics of Downsizing" *Datamation* October 15, 1992, pp. 107-110.

[11] *Status of the Department of Defense Corporate Information Management (CIM) Initiative*, October 1992 - reprint edition.