Reverse Engineering of Data

Peter Aiken, Ph. D. -- paiken@acm.org -- 804/828-0174
http://fast.to/peteraiken

Department of Information Systems
Virginia Commonwealth University

Office of the Chief Information Officer
Defense Information Systems Agency

Introduction

Interest in reverse engineering is growing as organizations attempt to reengineer existing systems instead of replacing them. When a system is reverse engineered, it is examined, documented, modeled, analyzed, and understood in order to better inform subsequent efforts. Of additional value, the reverse engineering analysis outputs can being reused as a source of enterprise architecture components. Since successful systems reengineering (SR) depends on effective reverse engineering, it (reverse engineering) is viewed as a critical part of SR.

Figure 1 **Data re-engineering taxonomy—adapted from Chikofsky and Cross.[1] (Note: The requirements outputs are optional—it is possible to proceed directly from the "as is" design assets to the "to be" design assets, bypassing the requirements assets when they are not necessary.)**

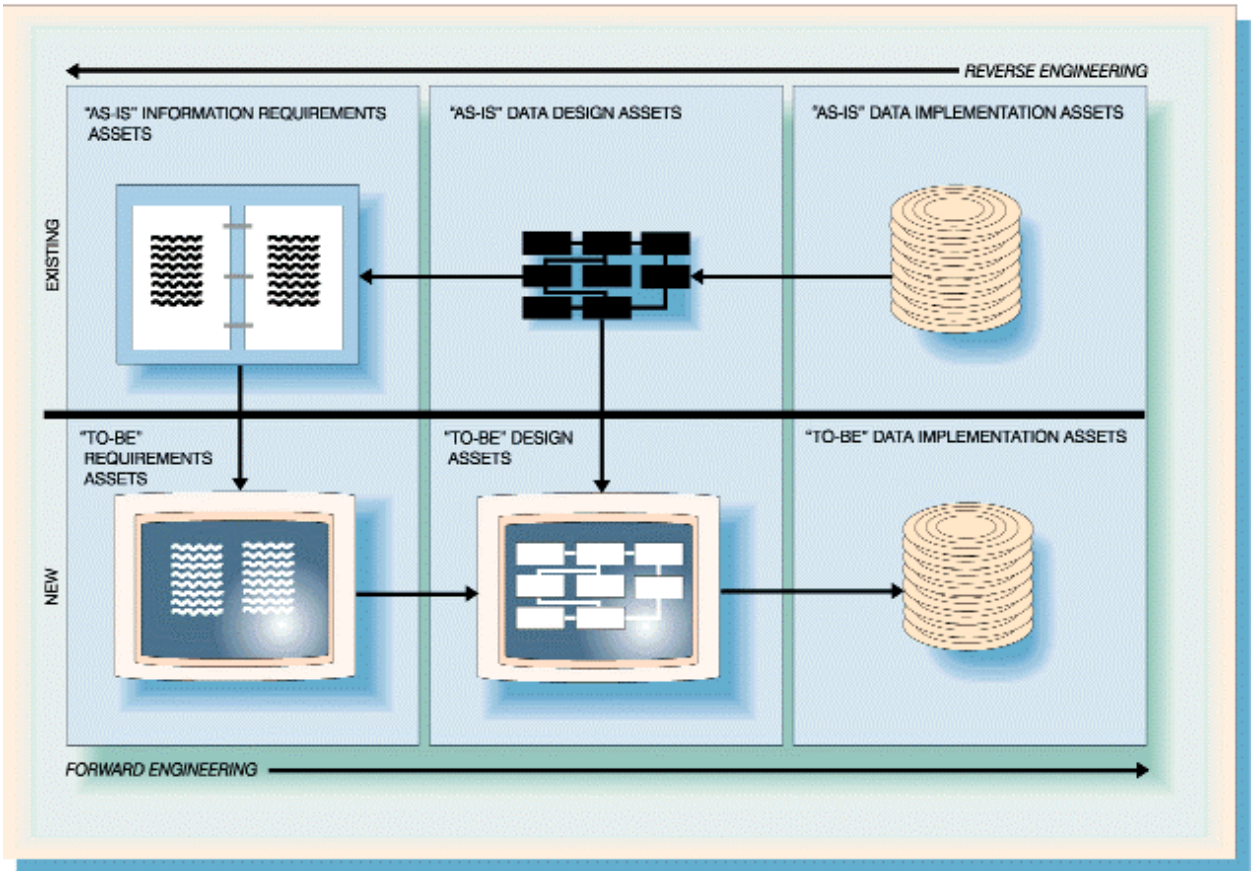
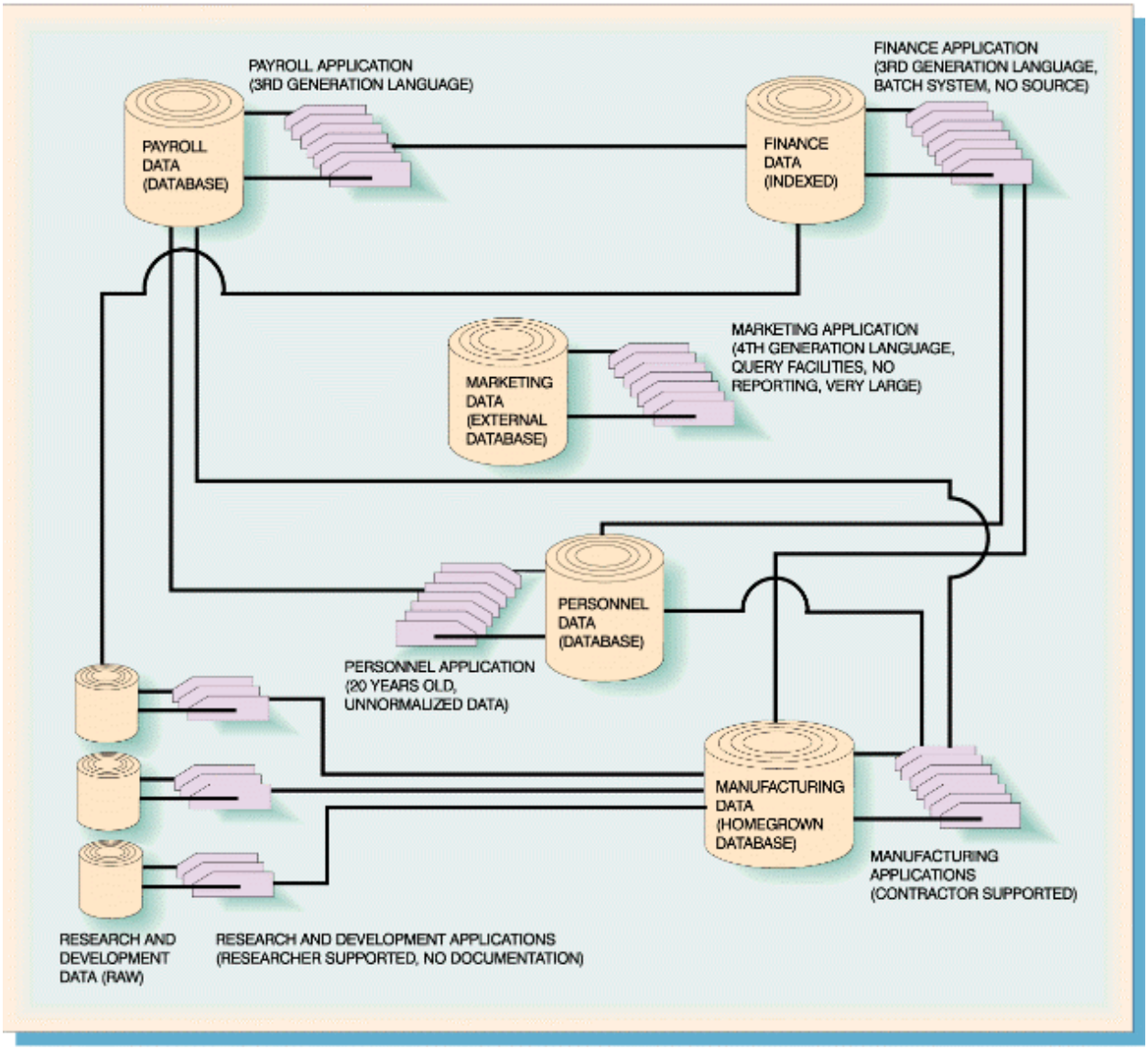


Figure 1 illustrates how SR is based on coordinated reverse and forward engineering activities, where forward engineering benefits from information gained by reverse engineering. A general SR goal is often stated as - delivering output meeting user requirements, using systems that currently do not. Organizations are turning to SR as a means of upgrading their existing information systems in situations where it appears to be a less expensive alternative to system replacement [1]. Three conditions can make it difficult for organizations to adapt their information systems to meet changing business needs - each is described below.

Figure 2 **A simplified version of Durell's[3] Gordian Knot with nine interfaces linking five stovepipe systems each supporting a functional area. (Note: Most functional business areas support multiple applications—increasing the actual number of interfaces proportionately.)**



- ***Complex Legacy Environments*** - many organizations have developed stand alone, or 'stovepiped' information systems (IS) that are both brittle and unintegrated. Over

time, changing user and business conditions continually evolve information integration requirements. Interfaces were typically developed in response, linking the outputs of one system to the inputs of another based on common understanding of the data. Interfaces describe the requirements for periodic data exchanges among systems. Eventually brittle situations, such the example shown in Figure 2, are the result. Because these systems weren't developed to easily exchange data, they don't. Changes, for example, in the payroll database require might require corresponding changes to personnel and manufacturing applications. Changes to the personnel applications might require corresponding changes to the personnel database that may in turn also require still further changes to the manufacturing applications, etc.

- ***Business Reengineering*** - Unintegrated and brittle information systems are also often barriers to a popular business process reengineering (BPR) technology - shared data. Data sharing can be defined as logically centralizing an integrated organizational data store containing integrated requirements that are capable of meeting multiple organizational and/or user requirements. Almost specified as universally as a 'user friendly interface,' the concept of ***shareable data*** is a key technological requirement for many BPR efforts (see [2], [3], or [4] for examples). Data sharing is prerequisite to organizational integration. Before efficiently sharing data across the organization and with external partners, organizations must analyze and integrate their data.

- ***Data Access Difficulties*** - in addition to being key to implementing many of today's business practices, data sharing difficulties within an organization can cause information to be difficult to obtain and expensive to maintain. These characteristics can effectively discourage sharing with external partners and block important growth opportunities. For example, organizations such as Wal-Mart prefer to conduct business with partners by directly exchanging data (see [5] and [6]).

Faced with these conditions, organizations are wondering where they should begin and what has worked for other organizations as they address problematic data issues? Reverse engineering a system's data has proven a successful approach to reconstituting the understanding and/or the physical condition of organizational data systems that have deteriorated or become unclear. The remainder of this paper describes the reverse engineering of data as applied to resolving organization data problems. It presents an overview of the reverse engineering of data, characterizing the current state of the practice and detailing an approach co-developed by the author. The next section of this paper defines the reverse engineering of data using a DRE template and a DRE activity model. The third section describes DRE guidance, analysis, and tools. The fourth describes situations when DRE has proven successful. The paper closes with a discussion of the lessons learned.

The Reverse Engineering of Data

Reverse engineering goals are: to analyze a system; to identify the system's components; to identify the system component's interrelationships; and to create representations of the system in another form or at a higher level of abstraction [7]. When considering reengineering as to system enhancement methodology, the question arises as to what reverse engineering techniques should be applied? Types of reverse engineering actively being researched include: system, software, database, and data.

An initial reverse engineering focus has been on software. The annual IEEE reverse engineering conferences have been populated with software oriented reverse engineering research, investigating topics such as the automation of techniques that answer questions such as: What does this program do ... ?" "Why does this program do ... ?" or "How does this program perform ... ?" [8]

If the focus of a reverse engineering effort is on system or organizational data, the analysis should be labeled as ***data reverse engineering*** (DRE) - defined as: the use of structured techniques to reconstitute the data assets of an existing system [9]. DRE offers an effective means of addressing situations where:

- the scope of the investigation is on the system wide use of data;
- the problem sparking the investigation is caused by problematic data exchange or interfaces; or
- the reengineering goals require a more strategic than operational analysis focus.

Consider as an example, a situation (described later as Scenario #1) with more than 1,400 application programs associated with the personnel system to be replaced. The functioning of just a few programs was of interest to the SR effort because the basics of personnel information management are generally understood (see for example, Figure 3.5 of [10]) and because the vast majority were being replaced by new system components. With the exception these few programs, individual program functionality was less important than understanding the system data oriented input, output, and maintenance capabilities. These were analyzed so that potential replacement system capabilities could be assessed for their ability to satisfy the current and future organizational requirements.

A further distinction can be drawn between the reverse engineering of data and the reverse engineering of databases. In a series of publications, Blaha (see for example [11], [12], or [13]) and others have described many aspects of database reverse engineering. Because, by definition, databases possess certain homogenous characteristics [14], database reverse engineering is often a more structured version of data reverse engineering. Often times the database schema, the metadata, the directory structure, or other system descriptions can be reported automatically, leading to reverse engineering activities that are more tightly focused with respect to the project duration, reverse engineering technique, and tool set. On the other hand, because of

greater likelihood of encountering non-standard systems, DRE tends to be potentially more involved, broader in scope, and less well supported from a tool perspective.

Another situation that required DRE analysis was characterized by a 'home grown - one of a kind' data management system utilizing fixed length 5,000 character records maintained by a system that serviced more than 100 different Federal agencies. The data structures of the individual records were translated at run time using a series of conceptual schema overlays. Any given record's layout was dependent on both its data content type and its agency affiliation in order to determine which overlay would correctly read the data. There was no chance of locating CASE tool support and the data engineers had none of the traditional database structure rules to rely upon when performing the analysis. Because of a low degree of automated support, DRE was accomplished manually by the data engineering team.

This illustrates where DRE was a cost effective, data centered approach to systems reengineering where automated techniques were not available or not materially useful. DRE provides a structure permitting data engineers to reconstitute specific organizational data requirements and then implement processes guiding their resolution. Because it is a relatively new formulation of systems reengineering technologies most organizations are unaware of DRE as a technique and practice a less structured approaches in response to data challenges. This variation of DRE was developed as an outcome of the Department of Defense's Corporate Information Management Initiative where the author's position as a reverse engineering program manager was to oversee the requirements engineering and formalization of thousands of management information systems requirements supporting Defense Operations [15].

In this and the following section DRE is described in more detail using: a DRE template; a DRE activity model; and a model of the data to be captured during DRE analysis - a DRE metadata model.

A DRE Analysis Template

Table 1 illustrates a DRE analysis template providing: a system of ideas for guiding DRE analysis; an overall metadata gathering strategy; a collection of measures; and an activity/phase structure that can be used to assess progress toward specific reengineering goals.

The template has been used to facilitate project knowledge development on a number of data reengineering projects. It consists of 13 activities, comprising three analysis phases: initiation, implementation, and wrap-up. A number of outputs are used to leverage subsequent system enhancement efforts. Each DRE activity produces a specific output and associated activity measures. Production and acceptance of outputs delivery signals activity completion. For example, the fifth activity, 'preliminary system survey,' results in the data contributing to the development of an analysis estimate. Estimate data establishes the analysis baseline and also produces an initial assessment of the analysis estimation process that can be periodically reexamined.

#	Activity	Outputs - Measures
	INITIATION PHASE	
1	Target system identification	Candidate target systems - the 'size and shape' of the reengineering challenges
2	Preliminary coordination	Target system points of contact - measures of non-technological complexity
3	Evidence identification & access	Target system evidence - indications of system reengineering feasibility
4	Analysis team initiation	Team directory - determination of ROI component derivations and starting date
5	Preliminary system survey	Analysis estimate data - analysis baseline and an estimation process evaluation
6	Analysis planning	Analysis plan - measure targets are established and initial gathering begins
7	Analysis kickoff	Analysis charter and authorization - articulation of initial: system; financial; and organizational expectations
	IMPLEMENTATION PHASE	
8	Target system analysis	MR/V cycles (repeat until complete) - cycle measures, team productivity data
8.1	— Cycle planning	Focused plan for next cycle - specific entities and attributes to be modeled
8.2	— Evidence acquisition	Structured evidence - indications of team cohesion and domain knowledge
8.3	— Evidence analysis	Candidate entities and attributes - TT data bank component
8.4	— Straw model development	Data entities organized into models - TT straw model development
8.5	— Model refinement and validation (MR/V)	Clearer, more accurate, validated models, TT model validation, RC to date and per session
8.6	— Model storage and organization	Accessible models and associated information, TT model storage, RC to accessibility, data engineer productivity data
	WRAP-UP PHASE	
9	Data asset packaging	CASE tool-based data assets - time stamp data, TT & RC per modeling cycle
10	Data asset integration	Integrated data assets - TT & RC to integration
11	Data asset transfer	Shared and shareable data assets - TT & RC to transfer
12	Analysis measures assessment	Additions to the analysis measures database - measures assessment
13	Template refinement	Continually improving template and implementation capabilities - analysis measures

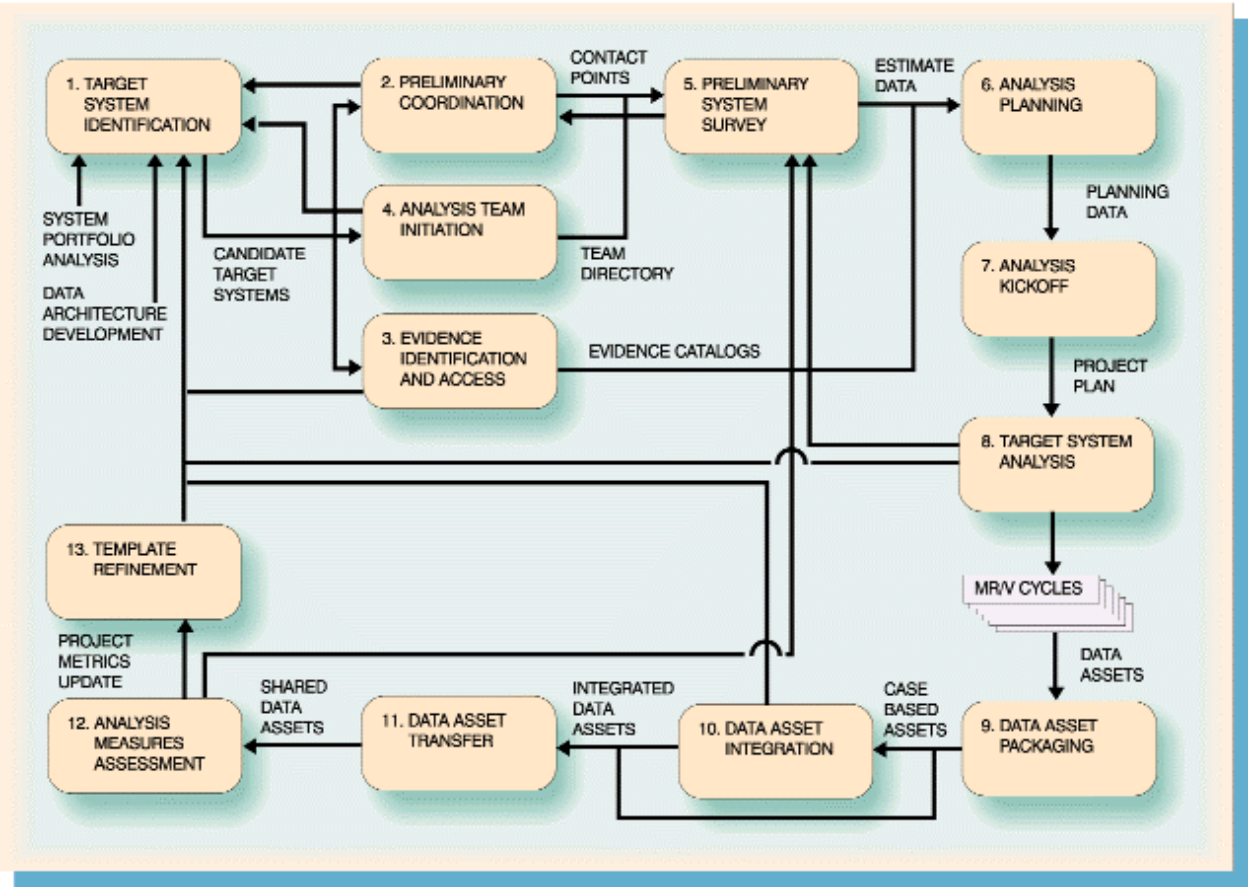
Table 1 DRE analysis template (TT = time to; RC = resources consumed).

In the next subsection, each template activity is described in the context of a DRE activity model.

DRE Activity Model

DRE analysis begins with typical problem solving activities. Initiation is concerned with identifying, understanding, and addressing any administrative, technical, and operational complexities. Initiation activities are designed to ensure only feasible analyses are attempted. Figuring prominently in the initiation phase is the development of baseline measures describing the reverse engineering analysis in conceptual size and complexity. These are used to develop an analysis plan. Figure 3 shows the template activities configured into a DRE activity model. Dotted lines illustrate potentially useful feedback loops among activities. Each activity is described below.

Figure 3 DRE activity model showing template inputs, activities, outputs, and feedback



Activity 1 - Target System Identification

The first DRE template activity is target system identification. The identification activity is required when organizational understanding of their data systems has degraded or become confused. Data architecture development activities guide, and systems performance characteristics motivate and inform target system identification. Activity 1 has two primary inputs. System performance data and, in particular, data on problematic system performance to help to identify specific data problems. Data architecture development needs can also influence the target system identification providing a second incentive to reverse engineer. This occurs when a DRE output contributes to the correction of a system problem and at the same time produces a lasting organizational data asset that assists in the development of an organizational data architecture component (illustrated subsequently in Figure 4).

Activity 2 - Preliminary Coordination

Since some systems are shared among organizational components with differing needs, the possibility exists for coordination difficulties. Preliminary coordination is required when systems serve multiple clients or when the reverse engineering can conflict with

forward engineering demands. In order to form the reverse engineering analysis team, it is crucial to secure management approval to access the skills and knowledge of available key system and functional specialists (a.k.a. key specialists). The cross-functional nature of DRE leads to three 'rules of thumb' for coordination:

1. Identified and prioritized system stake holder objectives must be synchronized with the DRE objectives and priorities.

2. DRE analysis cannot be successful without coordinated system management commitment. High-level management approval is necessary but not sufficient. Other management and systems personnel must also understand and support the DRE analysis objectives, or else organizational politics may jeopardize analysis success.

3. Negotiation, planning, and buy-in processes must be complete before attempting analysis.

Activity 3 - Evidence Identification and Access

Evidence identification and access has a broad definition. Obtaining access to evidence can range: from explicitly obtaining key specialist participation; to getting CASE tool readable versions of system dictionary data; to getting access to the proper versions of the system documentation. Data engineers assess the state of the evidence to estimate the effort required to develop a validated system model. Individual pieces of evidence can be classified as being in one of three possible states:

- ***Synchronized***. Synchronized evidence accurately represents the current state of the system. Synchronized is the most desirable evidence classification state. System documentation that is produced and maintained using CASE technology is most likely to be synchronized. It has been also, unfortunately, the rarest.

- ***Dated or otherwise of imperfect quality***. If documentation exists, it can be outdated or of poor quality. Dated system evidence reflects the system as it existed at a point in time. Changes have been made to the system since the evidence was created. Other types of data evidence imperfection could include corruption errors, technical errors, and value errors such as completeness, correctness, and currency [16]. This category describes most evidence available in DRE analysis.

- ***Not useful or not available***. The worst possible situation occurs when documentation was never created in the first place or has become subsequently not useful or is unavailable.

Activity 4 - Analysis Team Initiation

Initiation involves forming the analysis team, defining participation levels, and planning target system analysis. Team selection is important as members influence the

articulation of business requirements. Once constituted, beginning with the preliminary system survey, they collectively perform the remainder of the DRE analysis. To function effectively as a team, they need to understand the analysis goals in the context of an overall enterprise integration strategy.

Activity 5 - Preliminary System Survey

The preliminary system survey (PSS) is a scoping exercise designed to help assess the analysis characteristics for reengineering planning purposes. Survey data is used to develop activity estimates. The purpose of the PSS is to determine how long and how many resources will be required to reverse engineer the selected system components. The PSS is concerned with assessing system dimensions according to several types of criteria including the:

- condition of the evidence;
- data handling system, operating environment, and languages used;
- participation levels of key systems and functional personnel; and
- organization's previous experience with reverse engineering.

Completed PSS results provide system characteristics used to develop a sound cost-benefit analysis and a useful analysis plan. Two structured techniques are applied during the PSS: functional decomposition and initial data model decomposition. Each results in a validated model that serves specific roles (described in the next subsection). Model development produces data useful for estimating the remainder of the analysis. The models then guide subsequent target system analysis activities.

Activity 6 - Analysis Planning

Analysis planning involves determining: 1) key specialist availability; 2) the number of analysis team members; and 3) the number of weeks of analysis team effort. Core system business functions are evaluated for overall complexity, described using model components that are combined with a functional analysis rate per hour. The activity output is an estimate of the number of weeks required to accomplish the analysis. The team derives the analysis characteristics as a function of three components that are instantiated using organization specific data. Analysis characteristics are determined by the three components: the relative condition and amount of evidence; the combined data handling, operating environment, and language factor; and the combined key specialist participation and net automation impact component.

In general, the value of the term describing the combined data handling, operating environment, and language factor is greater than one. It serves as a confounding DRE characteristic, representing increased resources required to reverse engineer systems with obscure or unknown data handling, operating environment, or programming languages. This component typically increases the set of baseline characteristics established by the relative condition and amount of evidence component.

In contrast, the availability of key specialists and automation can significantly increase reverse engineering effectiveness. Thus the component value typically ranges between zero and one, reducing the overall analysis characteristics represented by the combination of the first two components. Once the analysis characteristics are known, the analysis estimate is determined as a function of the analysis characteristics and the historical organizational reverse engineering performance data (for more see [17]).

Activity 7 - Analysis Kickoff

Analysis kickoff marks the transition to implementation and the start of target system analysis. At this point it is useful to have achieved a number of setup milestones including:

- identified and implemented solutions to required coordination issues;
- educated colleagues and project team members;
- confirmed participation commitments; and
- achieved participant consensus as to the nature of the investment in this enterprise integration activity.

Activity 8 - Target System Analysis

Target system analysis is evolutionary in nature - modeling cycles are repeated until the analysis has achieved the desired results or (in some cases) the analysis has become infeasible. Modeling cycles use evidence analysis techniques to derive validated system models. This is the activity most conceptually associate with DRE analysis. It is focused primarily on correctly specifying (at the same or at a higher level of abstraction) information capable of describing:

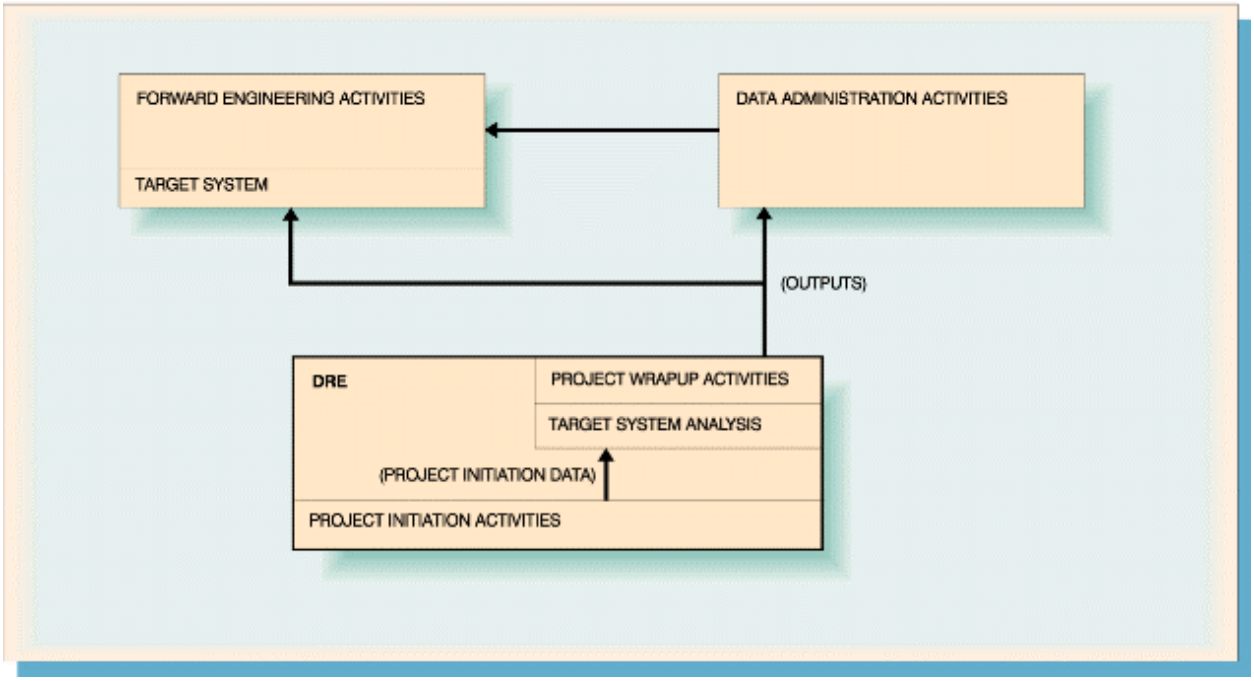
- ***System information connecting requirements*** - these are driven by the number of information sources and destinations; connecting in this context is defined as the ability to access data maintained elsewhere.

- ***System information sharing requirements*** - driven by the volume and complexity of the organizational information sharing and integration requirements, sharing is defined as the ability to integrate and exchange information across systems using a common basis for understanding of the data.

- ***System information structuring requirements*** - driven by the number and types of relationships between coordination elements, understanding system structures results in defined descriptions of user ability to extract meaning from data structures.

Target system analysis cycles are described in the next section.

Figure 4 Multiple uses of packaged DRE analysis outputs



Activity 9 - Data Asset Packaging

Figure 4 illustrates packaged data asset uses. Generally, data engineers complete activity 9. They supervise data asset validation, documentation, and packaging in usable and accessible formats. Data asset packaging ensures that data assets are correctly packaged for delivery to other enterprise integration activities.

Two output formats are particularly useful:

1. A usually paper-based format that the analysis team can point to and say something to the effect of 'the data assets created by this analysis are documented in this binder, and data administration can help you obtain electronic access to them.' While printed versions are largely symbolic, the value of packaged data assets is in its representation of the largely intangible analysis required to produce it.

2. An electronic, CASE tool-based format stewarded by the functional community and maintained by data administration. In organizations that have implemented CASE on an organization-wide basis, this information is readily accessible for other uses.

Because DRE analyses are made economically feasible by CASE tools, data asset packaging often occurs continuously as the validated data assets are developed and added to the data bank. When models are 'published' in the organizational data bank, they will be treated as organizational data assets facilitating and guiding future systems development.

Activity 10 - Data Asset Integration

Because of the cumulative nature of DRE analysis outputs, the data assets developed during DRE analyses can be made more valuable by integrating them with other data assets developed during other enterprise integration activities. Data asset integration involves, for example, explicitly addressing redundant data entities, data synonyms (where different terms have similar meanings), and data homonyms (same pronunciation but different meanings). This activity's goal is to resolve instances of data confusion and place the target system models in accurate perspective relative to other data assets. Outputs from activity 10 are integrated data assets. These assets are made more useful to the remainder of the organization through data administration programs.

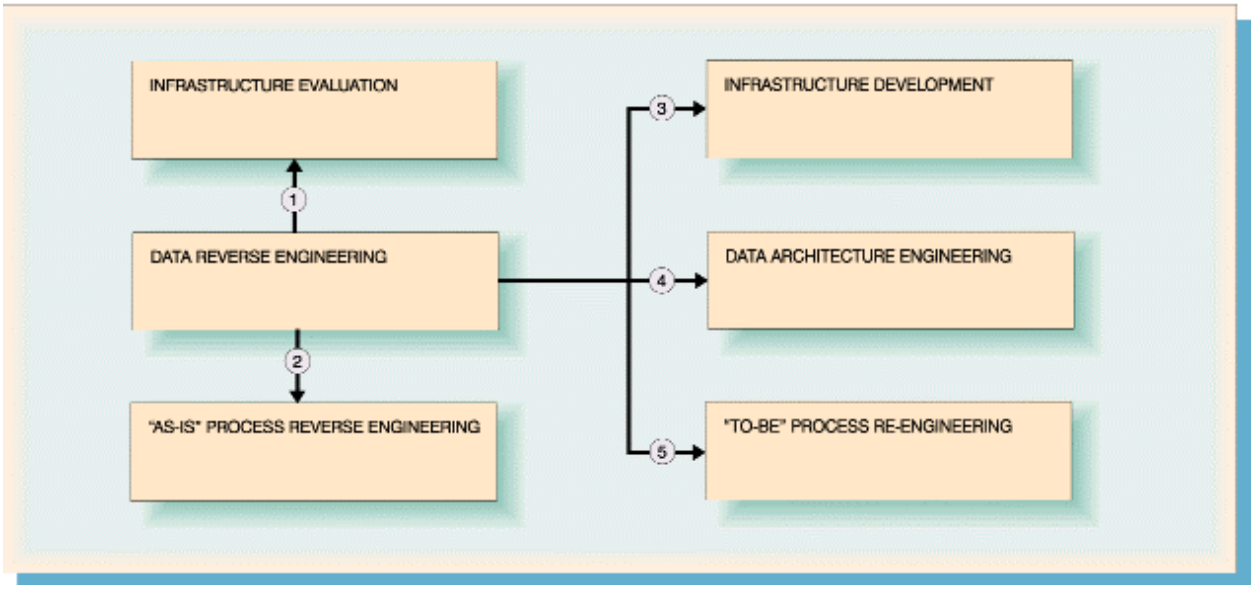
Activity 11 - Data Asset Transfer

Template activity 11 is formal recognition and enforcement of the fact that most DRE analyses produce outputs that are required by other enterprise integration activities. Making data assets available to other enterprise integration activities is the most tangible DRE analyses output. Data asset transfer enforces the notion that DRE activities are designed to provide specific information useful to other enterprise integration activities.

Figure 5 illustrates how a single DRE analysis can produce five different types of assets useful to other enterprise integration activities. Potential data asset transfers include the following:

1. Regular information exchanges with concurrent infrastructure evaluation activities help the organization to identify unmet gaps.

Figure 5 Outputs of a single DRE analysis can provide inputs useful to multiple enterprise integration activities



2. Data assets exchanged with As Is process reverse engineering efforts to concisely illustrate the existing organizational data capabilities.

3. System-related technology constraints and opportunities identified during DRE analysis often provide specific infrastructure requirements information to subsequent development activities.

4. Validated data assets are developed with the presumption that they will be integrated into the organizational data architecture.

5. An inventory of existing data assets, containing the type and form of current data can provide information about existing but unrealized data opportunities (such as mining). These can be quickly turned into 'low hanging fruit' in To Be business process reengineering activities.

Making data assets available can involve changing the media, location, and format of data assets to match requirements of other enterprise integration activities. For example, situations may arise where organizations are changing CASE tools. In these instances, the data assets may be translatable from one tool format to another via various import/export utilities and/or exchange formats. Other asset transfer requirements may occur when the enterprise-level models need to be extended to link to operational concepts or additional data assets. The outputs of activity 11 are data assets delivered on time, within budget, and meeting their intended purpose of proving useful as inputs to other enterprise integration activities.

Activity 12 - Analysis Measures Evaluation

After the analysis is complete, the team summarizes and evaluates the analysis measure data gathered periodically during the analysis. The evaluation is used to establish and refine organizational DRE productivity data used in both planning DRE and strategically assessing enterprise integration efforts. Examples of summary measures collected include:

- the number of data entities analyzed;
- the number of duplicate data entities eliminated;
- the number of shared data entities identified;
- the project rationale;
- the expected financial benefit;
- information describing the overall analysis throughput;
- assessment of the key specialist participation; and
- reactions of systems management to the analysis.

The outputs of activity 12 become another set of measurements in the overall enterprise integration analysis data collection.

Activity 13 - Template and Implementation Refinement

One of the most important analysis closure items is collecting and recording implementation measures, any refined procedures, tool and model usage data, and operational concepts. The outputs from template activity 13 are focused on assessing and improving both the template and subsequent implementation. The results and changes are archived to permit subsequent analysis.

The net worth of the analysis outputs often cannot be accurately evaluated immediately after the analysis. This is because the overall contribution of these outputs towards data administration goals and enterprise integration activities often become apparent only in the context of longer term reengineering activities. The nature of DRE analyses and all enterprise integration activities is such that the benefits increase in value as the results are integrated. DRE analysis should be periodically reviewed with hindsight to learn from the successes as well as the unexpected occurrences. The activity results are: improved procedures; data on tool and model usage; and the template implementation assessments.

DRE Guidance, Analysis, and Tools

As a structured technique, DRE analysis has three components: functional decomposition; data model decomposition; and target system analysis. Each is described below and should be applied using the following guidance:

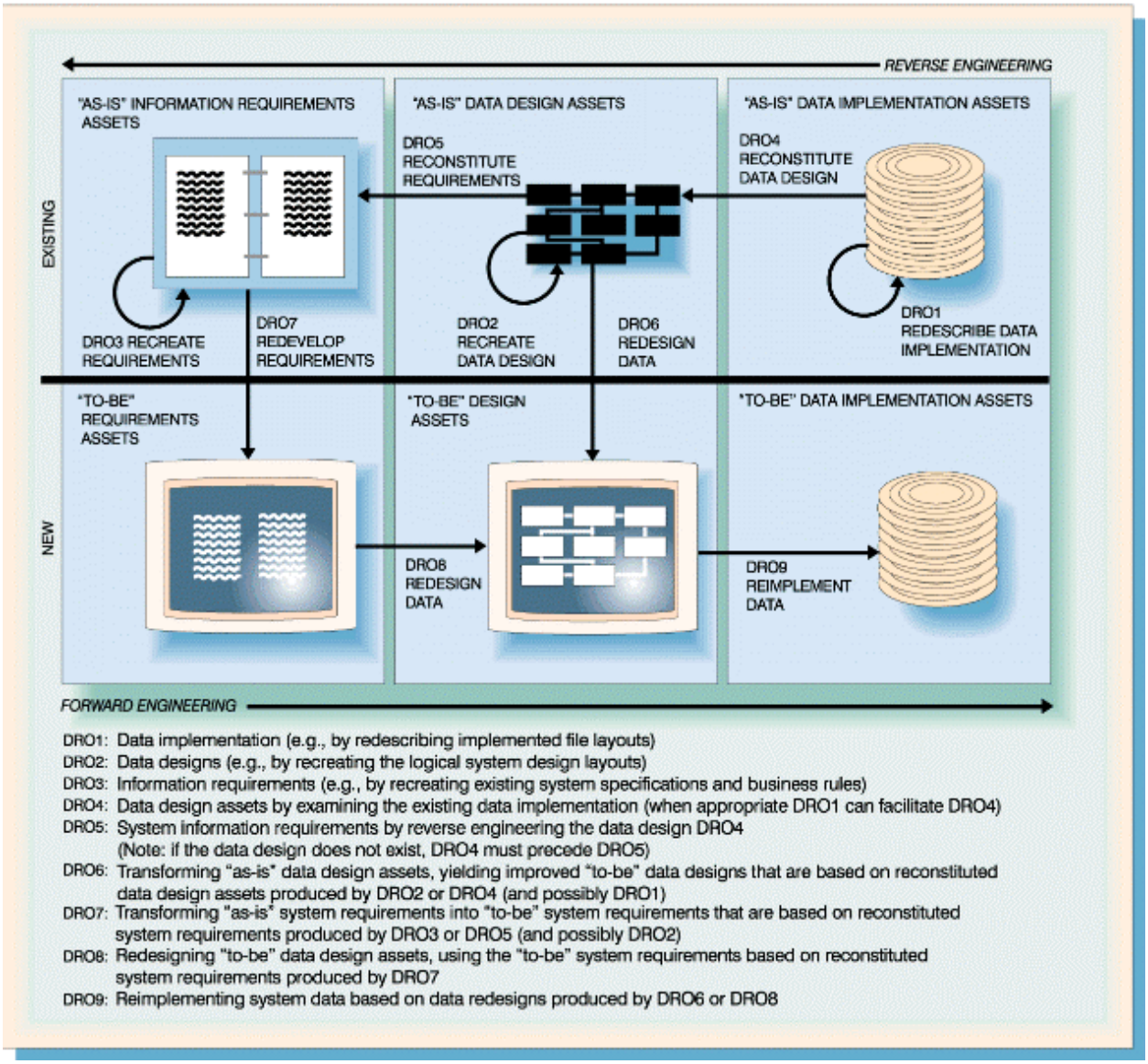
- ***Leverage of data management principles*** - understanding a relatively large amount of information by modeling and managing a relatively small amount of metadata. When scoping data reverse engineering projects, it is useful to understand nine possible types of data reengineering outputs (DRO) illustrated in Figure 6 and described below. Optimizing DRE projects involves identifying and developing requisite subsets of DRO.

- ***Modeling from integration points*** - unlike a jigsaw puzzle where it is important to begin at the edges, the structure of systems can often be understood most effectively by beginning with existing system interfaces and working into the system.

- ***Immediate rapid development*** - candidate (or straw) versions of the models developed early, quickly establish a common dialog among the analysis team and other involved personnel such as the customer. Because it is often easier to critique than to create, it is better to confront a key specialist with an imperfect model than with a blank screen.

- ***Living documents*** - by acknowledging that the models can be currently imperfect, the organization treats the models as living documents and that will evolve into more accurate versions throughout the analysis; this encourages constructive criticism from the collaborators and quickly draws newcomers into the process.

Figure 6 Data re-engineering taxonomy illustrating possible data reverse engineering outputs



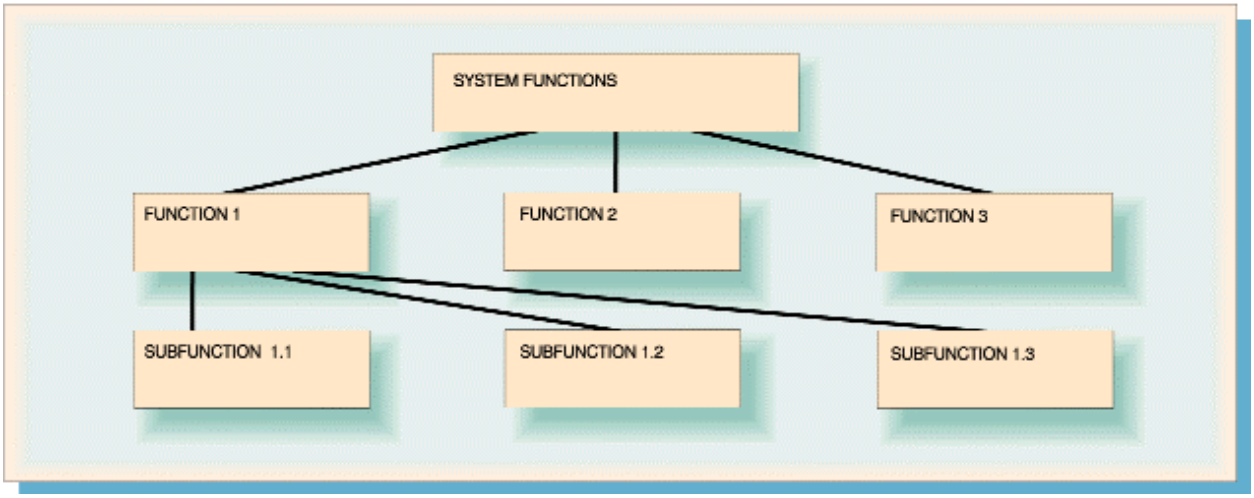
- ***Critical mass*** - understanding that the cumulative value of data assets increases at a more rapid rate as the degree of asset integration increases. The data assets produced are worth much more to an organization after they have been integrated with other data assets than by remaining as isolated groups describing individual systems or components. Over time, a key DRE goal is to expand the organizational knowledge structure with these data assets. As such, the relative value of the first data assets produced (or any single group of data assets) will be less that the value resulting from the integration of two or more data asset groups.

Functional Decomposition

Figure 7 shows a sample functional decomposition. DRE analysis develops an accurate functional decomposition of the target system. In some instances this already exists because it is a basic form of system documentation. When a valid system decomposition must be reconstituted, the analysis goal is to describe the system according to classes of related functions instead of attempting to deal with numerous, individual functions. Functional decompositions are usually maintained in the form of structure chart following standard diagramming conventions (e.g.; Yourdon, DeMarco, Gane and Sarson).

In a functional decomposition, the system is described in terms of the functions performed within single function (labeled System Functions). When accessing the electronic version, 'double-clicking' on System Functions reveals that it is comprised three primary functions. Each function can be further decomposed into subfunctions that can be further decomposed - down to the smallest useful description.

Figure 7 A sample DRE functional decomposition



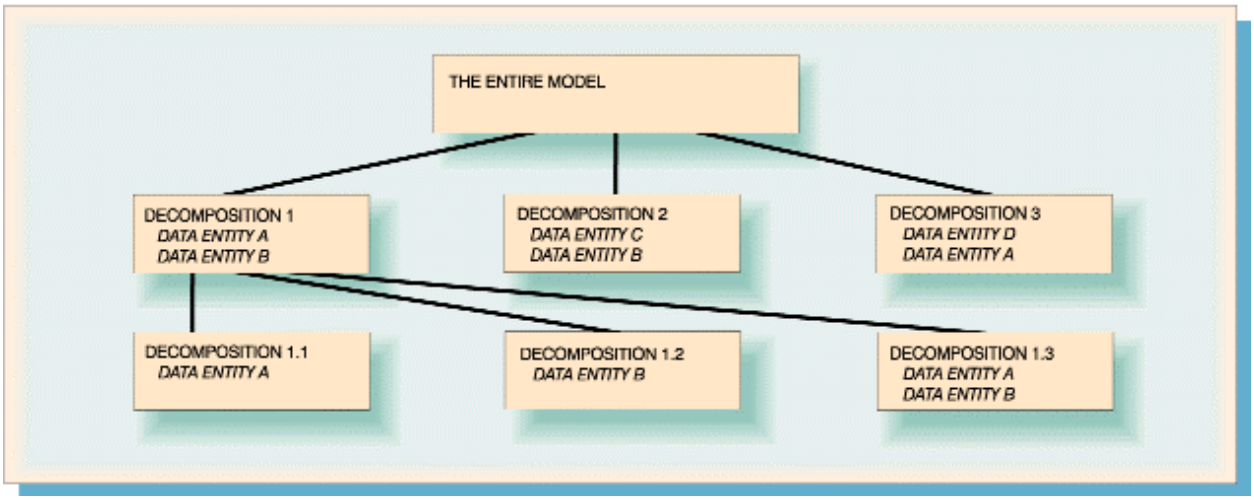
Unlike forward engineering, where analysts decomposes problems from the top-down, in reverse engineering, the decomposition is often constructed from the bottom-up by examining the system evidence. The answers are given and the question to be derived is - ***what sort of functions are performed by the existing system?*** If analysis resources permit, it can be cost-effective at this point to specifically identify subfunction data inputs and outputs permitting development of data flow diagram-type system representations. Collectively this information is used during analysis planning to establish milestones and to assess system size and complexity.

Data Model Decomposition

Figure 8 is an example of a data model decomposition. While similar in appearance, this model is used to maintain information associating groups of related entities, to each

other, and to categories of access (i.e. create, read, update, and delete) by certain groups of users. Using the functional decomposition as a basis, each unit of decomposition is examined to determine whether it constitutes a work group/collaboration focal point. The goal is to develop candidate arrangements of and then validated data entity groupings. Data model decomposition is accomplished by key specialists helping to model data relevant to each functional area represented by the data model components. Once validated, the data entity groupings are used to reassess the functional decomposition validity and as the basis for developing further project milestones. For some systems there will be high correspondence between the data model decomposition and the functional decomposition. For others, the data model decomposition will reveal different underlying data structures. In these instances the differences can be examined for possible process reengineering opportunities [18].

Figure 8 A sample data model decomposition



Target System Analysis - Data Reverse Engineering Metadata

Table 2 details the implementation phase (Phase II) of the template. Target system analysis consists of modeling cycles. Modeling cycle activities can occur in various formats ranging from: contemplative solitude; to phone consultation; to structured interviews; to evidence analysis; to JAD-like, model refinement-validation (MR/V) sessions.

The goal is to develop validated models of aspects of the target system. Candidate models are developed using: system data entities; the relationships between those entities; and organizational business rules. Candidate model development can be greatly aided by the use of available data model pattern templates (such as those catalogued by Hay in [12]).

Implementation phase			
#	**Name**	**Activity**	**Output**
8	Target system analysis (repeated until completed)		
8.1	Cycle planning	· Evaluating and incorporating previous cycle results · Identifying area of highest risk of lack of knowledge · Specifying analysis targets and a plan for the current modeling cycle	Focused plan for obtaining desired results from the next cycle
8.2	Evidence acquisition	· Collecting evidence · Cataloging evidence · Structuring evidence · Looking for missing evidence	Organized evidence
8.3	Evidence analysis	Analyzing evidence for appropriateness & model development potential	Candidate data entities
8.4	Straw model development	Creating candidate models	Data entities organized into models
8.5	Model validation / refinement	· Identifying changes in the model as a result of errors, new knowledge, and normalization · Documenting changes and further refining models · Validating models using appropriate techniques	Clearer, more comprehensive, more accurate, validated models
8.6	Model storage & organization	Collecting, cataloging, and structuring models for archival and configuration management purposes	Accessible models

Table 2 DRE Template Implementation Phase is comprised of modeling cycles.

The models are developed using system evidence (seven categories are defined in Table 3). The models are analyzed, reviewed, and improved by both functional and technical analysis team members. Revisions and refinements are made to the models as new or clarified information comes to light during these sessions. Data assets produced during DRE are stored in the organizational data bank along with other relevant analysis information. Model components are integrated with other components as required. When a critical mass or sufficient quantity of models have been integrated, the information in the data bank becomes capable of providing useful, consistent, and coherent information to all levels of organizational decision making, creating conditions for better organizational functioning.

Evidence category	**Examples**
Key specialists	Domain knowledge from the specialists, business rules
Processes	Functional descriptions, process models, code, user manuals
External data	Screen, report, interface specifications, interfaces to other systems
Conceptual data	Logical data models
Internal data	Program variables, data element lists, tables, file layout structures
Policies	Directives, guidelines, planning statements
System	Program source, object code, job procedures, libraries, directories, test cases, schemas, copylibs, make files, link maps, I/Os and other documentation, data

Table 3 System evidence categories.

Figure 9 Possible data bank users (adapted from Selkow[27])

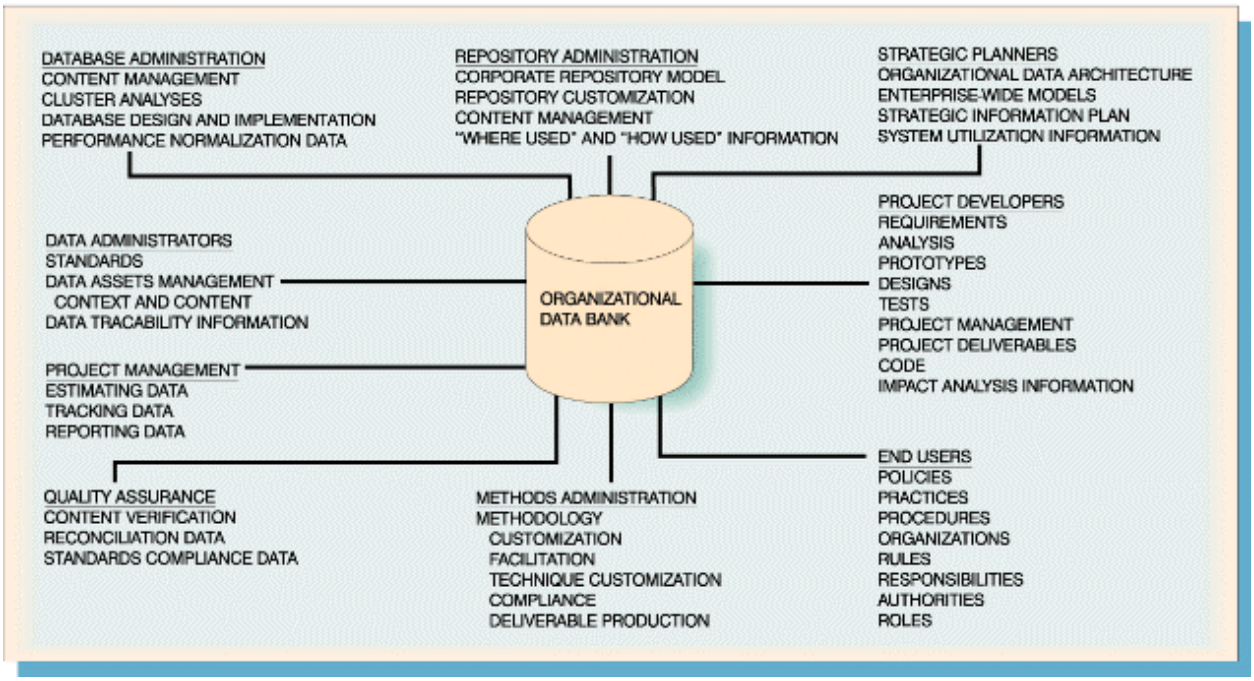


Figure 9 shows possible uses of DRE analysis information. DRE, and target system analysis in particular, focus on creating representations of the target system using appropriate entity relationship and other data modeling techniques. Figure 10 represents the data required for DRE as a metadata model - a model of the information capable of being captured during DRE (shown unnormalized to facilitate understanding).

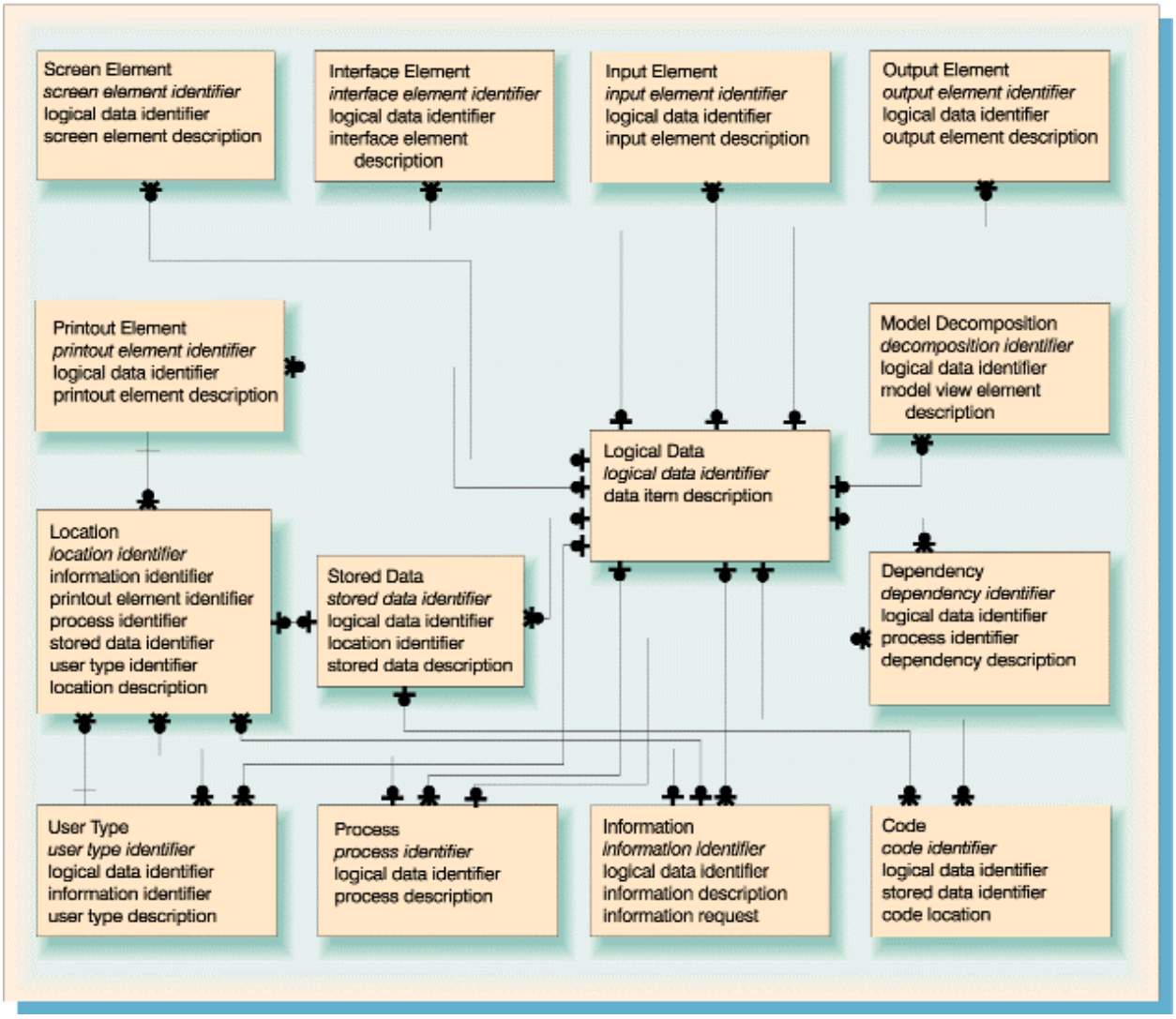
The DRE metadata model contains precise information required to understand the target system that was unavailable or disorganized before the analysis. Populating the DRE metadata model is the primary focus of target system analysis. The analysis goal is to produce validated metadata model data.

For example, the goal of the functional decomposition (described above) is to populate the PROCESS and DEPENDENCY entities. The goal of the data model decomposition analysis (also described previously) is to populate the initial version of the data stored in the LOGICAL DATA and MODEL DECOMPOSITION entities of the metadata model. Key to successful analysis planning is identifying just how much of the data is required in light of the analysis objectives. A description of the DRE metadata model follows.

Visually, the model is centered around the data entities: LOGICAL DATA and STORED DATA. LOGICAL DATA entities are the conceptual things about which a system tracks. Following standard definitions, LOGICAL DATA entities are facts about persons, places, or things about which the target system maintains information. Attributes are facts grouped as they uniquely describe LOGICAL DATA entities. Additional understanding is obtained from the way each entity is 'related' or not related to each other entity. STORED DATA entities are instances where a LOGICAL DATA entity is physically implemented. The

LOGICAL DATA entity on the other hand is populated with entity type descriptions. An association linking each STORED DATA entity to one LOGICAL DATA entity indicates that eventually one LOGICAL DATA entity should be related to one or more STORED DATA entities and each STORED DATA should be linked to at most one LOGICAL DATA entity. This structure indicates a requirement to define every PHYSICAL DATA entity by associating it with one LOGICAL DATA entity. Organization wide data sharing can begin when STORED DATA entities are commonly defined using LOGICAL DATA entity descriptions and applications process that data using the standard definitions. This mapping also permits programmatic control over the physical data using logical data manipulation.

Figure 10 **A DRE meta-data model showing *key* and other information that can be captured during DRE analysis. (Note: Many different types of attributes can be implemented for each entity—all represented with a single, nonkey entity description attribute.)**



Moving next to the upper left-hand corner, extending across the top row are four entities with the same association to the LOGICAL DATA entity. The entities SCREEN DATA,

INTERFACE DATA, INPUT, and OUTPUT also all have many to one associations between themselves and LOGICAL DATA. Information describing each CRT screen field is maintained using the SCREEN DATA entity. Each LOGICAL DATA entity is linked to every instance where system code causes the item to be displayed as a SCREEN DATA field. When populated, a database described by the model will maintain information as specific as: ***screen data attribute W of screen X is a display of attribute Y of logical data entity Z***. (Each attribute can be displayed in multiple places in the system.) The many to one pattern of association is repeated for the PRINTOUT, DATA MODEL DECOMPOSITION, DEPENDENCY, CODE, PROCESS, and LOCATION entities. The analysis goal is to be able to link each system INPUT, OUTPUT, INTERFACE, and SCREEN DATA entity, with one and only one specific LOGICAL DATA entity thus defining common use through out the system.

The MODEL DECOMPOSITION entity is associated with the LOGICAL DATA entity in a manner indicating that each LOGICAL DATA entity exists on one or more model DECOMPOSITIONS. (Recall from above that MODEL DECOMPOSITIONS are used to manage data model complexity by grouping data entities common to subsets of the overall model.) Following down the right hand side of the model, the DEPENDENCY entity is used to manage interdependencies for data entities that are derived from within the system and other functional or structural representations. At the bottom right corner is the entity CODE. CODE contains references to each of the system code locations that access each LOGICAL DATA entity. For example, something as specific as, ***data entity W is generated by a code location X of job-stream Y that is maintained at location Z.***

Moving to the left along the bottom row of entities, the model indicates that the entity INFORMATION has the same association but the interpretation here is definitional. INFORMATION is defined in terms of specific LOGICAL DATA entities provided in response to a request. Following Appleton [19] data is a stored combination of a fact and a meaning. An INFORMATION is at least one datum provided in response to a specific request. The request and any data provided in response are identified using the INFORMATION entity. The INFORMATION entity is also associated with one or more USER TYPES who generate specific information requests. In addition, INFORMATION is also associated with one or more specific LOCATIONS where the data needs to be delivered in order to be of maximum value. Similarly, an entity - PRINTOUT DATA ENTITY - accounts for printout elements. PRINTOUT is also associated with the LOCATION requiring the printout. The LOCATION entity has links to USER TYPES at a specific LOCATION, to the FUNCTIONS performed at that LOCATION, to the INFORMATION requested by that LOCATION, and to any system CODE stored at that LOCATION. FUNCTIONS are defined as the process of spending resources to deliver specific INFORMATION requested by a specific USER TYPE at a specific LOCATION.

Since target systems analysis is cyclical in nature, the focus is on evolving solutions from rapidly developed candidate or straw models that are refined with subsequent analysis. Three primary types of data are produced as a result: traceability information, the data entity definitions, and the data map of the existing system. While these three

are useful individually, they are made most useful when maintained in an integrated, CASE-based organizational data bank.

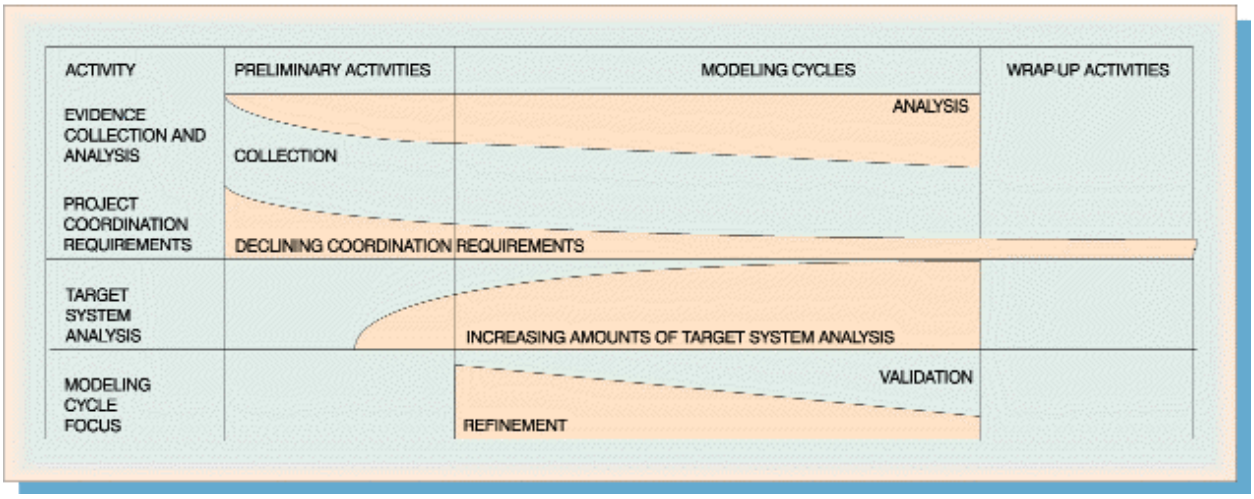
It is possible to accomplish target system analysis as a comprehensive examination of the system, beginning at one starting point and proceeding through the entire system. Usually this approach is unnecessarily cumbersome. Experience indicates that Pareto's Law applies to this situation - 80 percent of the time, DRE information requirements can be captured by focused analysis of 20 percent of the system. The question arises, how does the DRE team determine which is the 20 percent that they should focus on? This is guided in part by the scope of the models produced. Discrepancies between the functional decomposition and the data model decomposition should be targeted for early analysis to determine why the discrepancy exists between the users perception of the system functions and the data entity groupings in the system that (in theory) support the functions. Key is to not start at one 'edge' of the system and plan to work through the entire system in a comprehensive manner until the DRE metadata model is complete. Instead allow the DRE analysis goals to determine what information is required for the analysis, target specific system aspects, and model these within their operational context. Data from this analysis is used to populate appropriate portions of the DRE metadata model and develop products capable of meeting analysis goals. Consider it an exercise of knowing the answers and determining the questions.

Developing and maintaining the completeness of the traceability matrices as specified by the DRE metadata model is an important and challenging task. Since few CASE tools are capable of maintaining all of the required associations, organizations have been developing their own metadata management support using, for example, combinations of spreadsheet, word processing, and database technologies. As organizations become more proficient at DRE, the utility and ease of developing and maintaining the metadata will increase. Many CASE environments support data definition language (DDL) production as a modeling outcome, permitting rapid development by evolutionary prototyping of components such as: database structures; views; screens; etc.

The data bank is used to maintain all of the information in the DRE metadata model. It contains entity definitions stored as part of the corresponding data map. Key here is to map system components directly onto the metadata. The data model components derived from the system evidence are analyzed and entered into the CASE tool. The data map is constructed by defining and associating the data entity groupings identified as part of the PSS. Each data model decomposition is populated with attributes including key information. As these are developed, they are assessed against existing system data entities to see if they match. Aliases are also catalogued and tracked.

Four specific changes in the modeling cycle activities should be observed during DRE analysis. Figure 11 shows how the relative amounts of time allocated to each task during the modeling cycle change over time. It also illustrates how the preliminary activities occur prior to the start of the first modeling cycle in order to obtain the PSS information. The modeling cycle activity changes include:

Figure 11 Relative use of time allocated among tasks during DRE analysis



- ***Documentation collection and analysis***. Over time the focus shifts from evidence collection to evidence analysis.
- ***Preliminary coordination requirements***. Coordination requirements can be particularly high in situations where managers are unaware of the analysis context or the target system's role in enterprise integration activities. Once target system analysis commences, coordination requirements should diminish significantly.

- ***Target system analysis***. Just as the documentation and collection and preliminary coordination activities decrease, the amount of effort that can be devoted to target system analysis should increase steadily - shifting away from collection activities and toward analysis activities.

- ***Modeling cycle focus***. By performing a little more validation and less refinement each session, the focus of modeling cycles shifts correspondingly away from refinement and toward validation activities.

The purpose of DRE analysis is to develop models matching the existing system state. Model components should generally correspond one-for-one with the system components. Normalization and other forms of data analysis are deferred to forward engineering activities and are performed on a copy of the models used to maintain the existing target system metadata. Additional information collected during this activity can facilitate the development of distributed system specifications. For example, sixteen additional metadata entities useful in planning distributed systems and obtainable as part of reverse engineering analysis are described in a later section (see Table 4).

Situations When DRE Has Proven Successful

This section presents several scenarios illustrating how DRE analysis has proven successful solving data problems. An interesting observation is that while DRE was developed as a part of system reengineering, it has been effectively applied outside of

that context (as in Y2K analyses). Scenarios illustrate how DRE analysis was used to recover system metadata that was used to solve a specific organizational data problem.

Scenario #1 - Distributed Systems Architecture Specification

To better meet evolving customer requirements, a system manager plans to evolve two existing legacy applications from a mainframe base, by combining them into a single, integrated two tiered and then to a three-tier client/server system. The multi-year plan is guided by an evolving, integrated data reengineering effort. DRE formed the basis for the data migration plans transforming the original systems to the two tier architecture. The integrated models were developed by reverse engineering the two existing systems.

The functional decomposition and data model decomposition assisted in the development of specific data model views that were prototyped with the various user communities, again, using CASE tool-based DDL output. The integrated data model consists of 126 entities and more than 2,800 attributes. The completed two-tier implementation consisted of more than 1,500 Oracle tables. When the planning for the two tier implementation was completed, the data engineers returned to modeling, this time to populate the INFORMATION, LOCATION and USER TYPE entities, supplementing the metadata with sixteen additional attributes (see Table 12).

These extended data models are being used as the basis to develop data architecture specifications for the three tier target system architecture. Subsequent analysis described specific user classes possessing different information requirement types at hundreds of different locations. Understanding the distributed information requirements by location, will result in mapping of data between users and information requirements, and it will become key, strategic system planning elements. (More information on this project is available, describing the data reengineering [20], business process reengineering integration [21], development of the metadata [22], and project decision analysis context [23]).

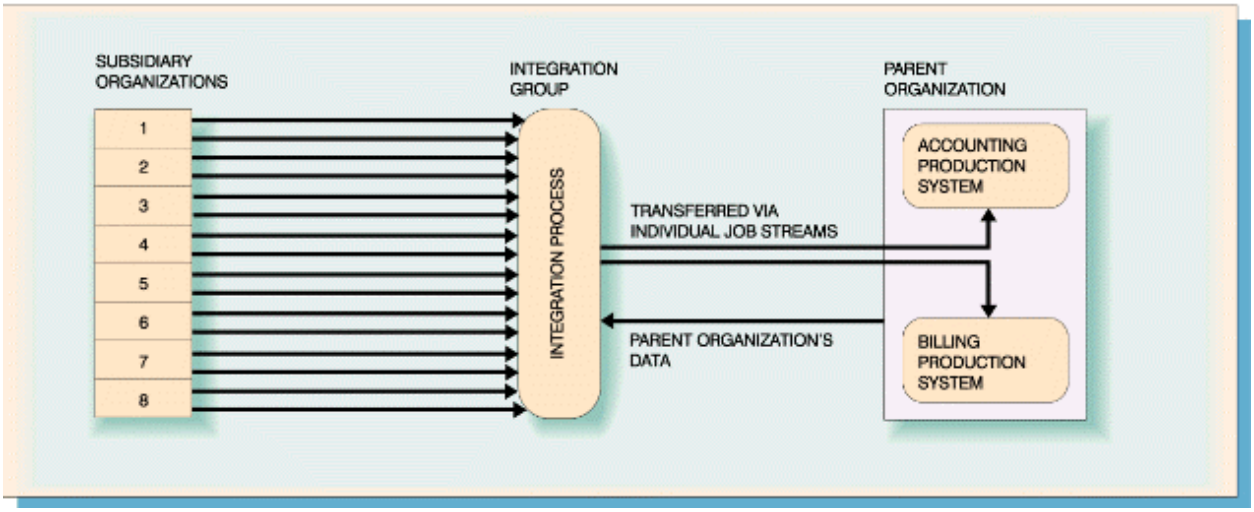
#	Primary use	Metadatum	Domain value
1	Architectural development	Data quality type	- Public - Private
2		Data usage type	- Operational - DSS I: single server node - DSS II: multiple server nodes - DSS III: distributed data and processing
3		Data residency type	- Functional - Subject - Geographic - Other
4	Application development	Archival data type	- Continuous - Event-discrete - Periodic-discrete
5		Data granularity type	Smallest unit of addressable data is an - Attribute - Element - Some other unit of measure
6	Architectural development	Data performance issues	Performance measurement/period of measurement
7	Application development	Data access frequency	Accesses/period of measurement
8		Data update frequency	Update/period of measurement
9		Data access probability	Likeliness that an individual data element of the total population will be accessed during a processing period
10		Data update probability	Likeliness that an individual data element of the total population will be updated during a processing period
11	Architectural development	Data integration requirements	Number and possible classes of nodes
12		Data subject area	Number and possible subject area breakdowns
13		Data grouping	Useful for cataloging user-defined clusters
14	Application development	Data location	Number and availability of possible node locations
15		Data stewardship	Range of all business units
16		Data system of record	System responsible for maintaining data element's data

Table 4 Metadata attributes useful in client server application development that are obtainable as part of DRE analysis (metadata derived from [24]).

Scenario #2 - Data Integration Problems

In a series of acquisitions, eight utility companies were merged with a parent organization. A data integration group was established to organize and produce job streams from the eight subsidiaries' data. Each of the eight subsidiaries transferred separate billing and accounting data to the integration group. The integration group's mission was to consolidate subsidiary with the parent organization's data, and remove errors from the job streams prior to transfer to the production systems (Figure 12).

Figure 12 **Data integration context diagram indicating the requirements to modify data to conform to expected system requirements, consolidate into a whole job stream, and perform a series of edit checks in preparation for transfer to the production systems**



Encoding it so it could be traced back to the originating system, the integration group performed a lengthy list of edit checks on the incoming data. When the data was thought ready, the integration group transferred the now scrubbed data via a job stream interface to the parent organization's production systems. The production systems responded to bad data by failing. All the data for an entire cycle must run at once, completely and without data errors in order to produce any output. In spite of rigorous scrubbing, repeated problems have cost significant resources to correct as both systems repeatedly fail due to bad or missing data. A puzzling characteristic was that no two problems encountered seemed the same - a unique data problem apparently produced each failure.

The solution was to focus the DRE analysis on the data crossing the interface to the production systems and work backwards into the integration group processing. A PSS determined the analysis challenge and established baseline measures. The LOGICAL DATA, STORED DATA, and INTERFACE DATA entities were modeled. The models became a systematized data asset, formally describing the production system data input requirements and permitting systematic analysis of each subsidiary's data. These models provided the starting point for further analysis and discussion between these organizations. Each subsidiary organization's individual data streams were systematically compared to the modeled interface data specifications. The previous practice had been to correct each data error in subsidiary data input streams.

Once populated, the DRE metadata model permitted programmatic data protection and maintainability. Delays associated with the accounting and billing production were reduced to the point where the integration group was no longer needed. The organization chose to reuse their experience to help reengineer other systems.

Scenario #3 - Developing Data Migration Strategies

A public sector run mainframe-based system was to be upgraded. The custom developed application served an entire functional area and contained program elements more than 20 years old. While the system functioned correctly and effectively, only two individuals in the organization understood the structure of its home grown, data management system. Fixed length, five thousand character records were coded, linked, and composed using thousands of different combinations to maintain data for many different organizations. The government funded an upgrade to replace the data management system. A question was raised as to the new data management system. Some argued for a relational database management system for maximum data flexibility. Others claimed the anticipated query volume would be too great for a relational implementation and insisted alternate models were more appropriate.

The solution was developed by formally modeling the existing system data as part of the data migration planning. The PSS determined that almost one hundred different functions were embedded in the system - leading to the development of a corresponding model decomposition. The PSS also indicated the analysis would require a six person analysis team, two months to complete the model. PSS results directed analysis to populate the metadata model with information linking LOGICAL DATA to SCREEN DATA, to PRINTOUT DATA, and to INTERFACE DATA entities. More than 500 STORED DATA entities were linked via LOGICAL DATA entities to 100 key reports, screens, and interfaces. A set of 200 LOGICAL DATA entities were documented. The completed model also documented more than two hundred business rules. It was determined that the query volume could be reduced to 20% of the original by developing a separate data warehouse permitting intranet access to typically requested information that would be extracted periodically from operational data. Based on this system design, a relational database management system was selected as the new data management system. The team used a CASE tool to maintain the required analysis data including the system data model and analysis data dictionaries.

Scenario #4 - Improving System Maintenance with CASE

As a result of a merger, a new work group was established to perform maintenance on a 1960's vintage application system. In the mid 1980's, a consulting partner introduced CASE technology as part of a co-development situation. The partnership failed, the employees who had been trained in the use of the CASE tool were downsized, and the system documentation was not kept synchronized with maintenance application. The new work group wanted to quickly become knowledgeable about the system and was also CASE illiterate. The team leader decided to address both issues simultaneously, and acquired the most recent version of the CASE tool. Next step was to develop a CASE training program for the work group that focused on recovering the system data assets using the CASE tool. This tool supported automated development of data models from existing physical data structures by importing the schemas into the tool. Much of the DRE metadata model was quickly populated by the work group as part of the training exercise including the PRINTOUT, SCREEN, INTERFACE, INPUT, OUTPUT, and

STORED DATA ELEMENTS. From these, the system functional decomposition and data model decompositions were developed, as was the system data dictionary and data map -- the organization had never previously developed this form of system documentation. This information was compared to the most recent system documentation. Several things became possible once the work group had accurately reconstituted the system documentation. The work group:

- Developed a much better system understanding as a result of the DRE-based CASE training exercises.
- Increased its effectiveness estimating proposed system changes due to better understanding of the system models.
- Became more effective in system maintenance application as a result of greater familiarity with system component interactions.
- Gained more knowledge as to how the system fit into the larger organizational information processing strategy.
- Was increasingly consulted for advice on data problem correction, functioning as an organizational reengineering resource.

Under these circumstances, the solution was found in the synergy between system maintenance application and developing work group CASE tool experience. Using the CASE tool's ability to programmatically reverse engineer the system data, the team used their growing DRE knowledge of the system to facilitate CASE tool understanding and vice versa. By reverse engineering the data using a CASE tool, the work group became more knowledgeable of both the case technology and the system itself as part of the same exercise. In recognition of increased work group performance, members were asked to become first consultants and then data engineers on other analyses.

Year 2000 Analyses

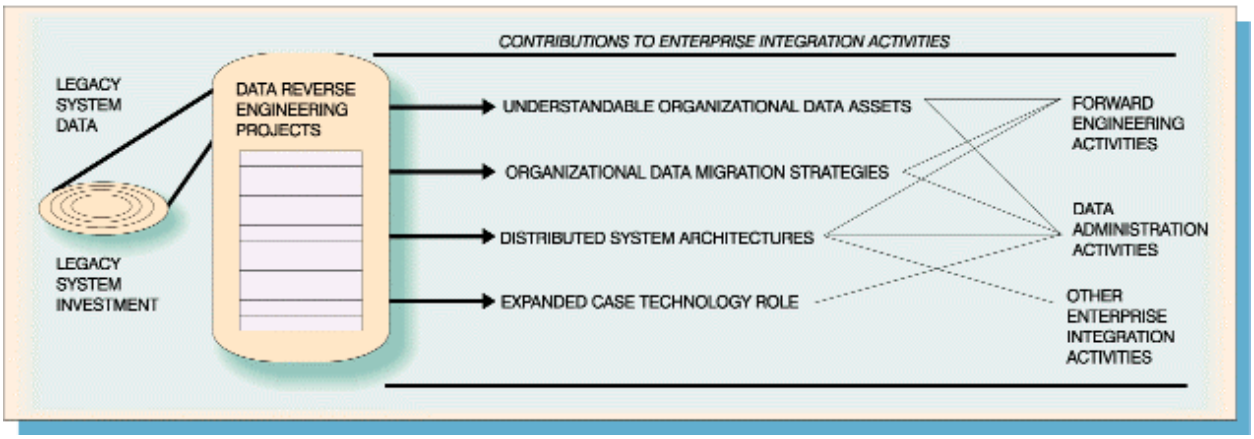
DRE has an obvious application as part of structured means of addressing organizational year two thousand (Y2K) data problems, easily providing structure for Y2K investigations. Through DRE analysis well prepares organizations to open for business on Monday - January 3rd, 2000. Target system analysis can be highly correlated with the activities performed as part of organizational year two thousand (Y2K) analyses. If approached from a DRE perspective, during target system analysis, date oriented or derived data can be flagged for further, Y2K specific analysis. If approached from the Y2K perspective, the examination and confirmation of Y2K compliance can be accomplished by storing the data elements in a CASE repository. Consider potential precision when implementing Y2K fixes by consulting validated DRE metadata (for example: maintaining information such as the names and locations of all code accessing the variable 'year' or the locations of all date based calculations).

Lessons Learned

Data reverse engineering represents an emerging technology with capabilities to serve multiple organizational roles. Particularly in systems reengineering contexts, it can be

used by managers interested aligning their existing information systems assets with organizational strategies to accomplish more effective systems reengineering. Selectively applied, DRE can also be an important first step toward increased organizational integration. Data-based success stories such as AT&T's entry into the credit card business, MCI's Friends and Family program, and the airline industry's systems supporting reservation and frequent-flyer programs, demonstrate the value of capitalizing on organizational data to implement successful business strategies (see [25]). Management is becoming aware of the true value of their data as an organizational resource, ranking it #2 (behind 'organizational architecture development' and in front of 'strategic planning') in a survey of 1990s MIS management issues [26]. Figure 13 illustrates that DRE analysis outputs describing an existing system can be used as a common source from which other enterprise integration activities result.

Figure 13 A DRE template used to provide a basis for other enterprise integration activities



The 1997 reengineering market is estimated to be $52 billion - with $40 billion to be spent on systems reengineering [27]. Understanding how DRE can provide a basis for other enterprise integration efforts prepares managers to recognize conditions favorable to its successful utilization. To cite an instance, the project manager in Scenario #1 realized the value of reverse engineering his two existing systems and subsequently directed our research team to reverse engineer the newly delivered, widely installed, commercial software application system in the belief that the effort would also be productive (see [28] for details). In this instance, the exercise achieved four primary organizational incentives for data reverse engineering within the project context:

1. Bringing under control and directing the organizational data assets for integration and sharing;

2. Identifying data migration strategies by understanding existing organizational information requirements and developing corresponding data migration plans;

3. Providing an information base for use in developing distributed, system architectures capable of meeting organizational needs; and

4. Expanding the role of CASE-based technologies within the organization beyond their traditional role in new systems development.

A future data reverse engineering research agenda includes investigation into additional system metadata uses. Leveraging metadata can contribute to other enterprise integration activities including:

- ***Integration with object modeling***. The reverse engineering metadata can be used as the basis for organizationally evolving or transitioning to an object orientation. The capabilities of CASE Tools capable of integrating object and data metadata will be subject of research investigations.
- ***Development of 'common use metadata.'*** If it is possible to define common use metadata, the research focus can shift away from understanding the metadata contents and towards metadata use by application developers building on current repository technology sought after by Microsoft [29] and others with metadata standardization projects.
- ***Expert systems.*** Incorporation of organizational expertise into metadata presents an intriguing challenge. Future research plans include examining the degree to which the organizational metadata can provide expert system-based advise on human resource policy implementation.
- ***Data Warehouse Engineering.*** In scenario #1, the project manager doesn't have the resources to rebuild the data warehouse - it is a situation where it must be implemented correctly the first time. Effective metadata use is required to correctly engineer data warehouses [30].

Acknowledgments

Much of the content for this paper was prepared in response to an invitation to address a group of information system managers in the Netherlands in early 1996 on the subject of legacy systems reengineering organized by Dr. Volken de Jong of Origin Groningen. The paper benefited enormously from critiques by several of my colleagues within the US Department of Defense - Data Administration Program and the comments of four insightful but anonymous reviewers for the ***IBM Systems Journal***. The DRE metadata model was developed using Visible Advantage™ from Visible Systems Corporation of Waltham, MA. The underlying research was sponsored in part by the Virginia Department of Personnel & Training. Bill Girling, Manager of Systems for the Department saw the need and developed the initiative. Many classes of data engineers have worked on DRE projects as part of their curricula in information systems in the School of Business. In particular, I'd like to thank the Information Systems Research Institute research associates Kim Boos, Leslie Borman, Lewis Broome, Sasipa Chankaoropkhun, Pawan Pavichitr, and Sirirat Taweewattanaprecha, for their professional contributions to these projects.

References

1 R. N. Karr Jr. ***Data Management Issues Associated with Stovepipe Systems*** General Services Administration/Information Resources Management Service/Policy Analysis Division, October 1993, KMP-94-1-I.

2 D. Stoddard and C. J. Meadows "Capital Holding Corporation-Reengineering the Direct Holding Group" (case in) J. Cash, R. Eccles, N. Nohria, and R. Nolan ***Building the Information-Age Organization: Structure, Control, and Information Technologies*** Irwin 1994, pp. 433-452. Boston, MA

3 J. Raymond Caron, Sirkka L. Jarvenpaa and Donna B. Stoddard “Business Reengineering at CIGNA Corporation: Experiences and Lessons Learned from the First Five Years” ***Management Information Systems Quarterly***/September 1994 18(3):233-250.

4 ***Beyond the Basics of Reengineering: Survival Tactics for the ‘90s*** Quality Resources/The Kraus Organization Industrial Engineering and Management Press, Institute of Industrial Engineers, Norcross, Georgia 1994.

5 L. Wilson "Cautious Change for Retailers" ***InformationWeek*** October 10, 1994, pp. 177-180.

6 S. Haeckel & R. Nolan "Managing by wire" ***Harvard Business Review*** September/October 1993, 71(5):122-132.

7 E.J. Chikofsky "The database as a business road map" ***Database Programming and Design*** May 90 3(5):62-67.

8 ***Proceedings of the First Working Conference on Reverse Engineering*** May 21-23, 1993, Baltimore, MD 233 pages.

Proceedings of the Second Working Conference on Reverse Engineering IEEE Computer Society Press Toronto, Ontario, Canada July 14-16, 1995 335 pages.

Proceedings of the 3rd Working Conference on Reverse Engineering (WCRE '96) November 8-10, 1996 — Monterey, CA 312 pages.

Proceedings of the Fourth Working Conference on Reverse Engineering (WCRE '97) October 6-8, 1997, Amsterdam, The Netherlands 248 pages.

9 P. Aiken ***Data Reverse Engineering: Slaying the Legacy Dragon*** McGraw-Hill 1996.

10 D. Hay ***Data Model Patterns: Conventions of Thought*** Dorset House Publishing, 1995 p 23.

11 W. Premerlani and M. Blaha "An Approach to Reverse Engineering of Relational Databases" ***Communications of the ACM*** May 1994 37(5):42-49, 134.

12 M. Blaha "Observed Idiosyncracies of relational database designs" ***Proceedings of the Second Working Conference on Reverse Engineering (WCRE '95)*** July 1995, Toronto Canada pp. 116-125.

13 M. Blaha "Dimensions of Relational Database Reverse Engineering" ***Proceedings of the Fourth Working Conference on Reverse Engineering (WCRE '97)*** October 6-8, 1997, Amsterdam, The Netherlands pp. 176-183.

14 E.F. Codd "Relational Database: A Practical Foundation for Productivity" ***Communications of the ACM*** February 1982 25(2):109-117.

15 P. Aiken, A. Muntz, and R. Richards "DoD Legacy Systems: Reverse Engineering Data Requirements" ***Communications of the ACM*** May 1994 37(5):26-41.

16 Y. Yoon, P. Aiken & T. Guimaraes "Defining Data Quality Metadata: Toward A Life Cycle Approach to Data Quality Engineering" under review by the ***Information Resources Management Journal***.

17 P. Aiken & P. Piper "Estimating Data Reverse Engineering Projects" ***Proceedings of the 5th annual Systems Reengineering Workshop*** (Johns Hopkins University Applied Physics Laboratory Research Center Report RSI-95-001) February 7-9, 1995, Monterey CA, pp. 133-145.

18 P. Aiken & L. Hodgson "Synergistic Depencencies Between Analysis Techniques" in the ***Proceedings of the 1997 Software Technology Conference***, April 25-May 2, 1997 Salt Lake City, Utah.

19 D. Appleton "Business Rules: The Missing Link" ***Datamation*** October 1984, 30(16):145-150.

20 P. Aiken & B. Girling "Data Reengineering Fits the Bill" ***InformationWEEK***, May 26, 1997, pp 8A-12A.

21 L. Hodgson & P. Aiken "Synergistic Dependence Between Analysis Techniques" ***Proceedings of the 9th Software Technology Conference*** in Salt Lake City, UT - April 27-May 2, 1997, published on CD-ROM by the Air Force Software Technology Support Center http://www.stsc.hill.af.mil or 801-775-5555.

22 P. Aiken "Integrating the Reverse and Forward Engineering of Data Using Metadata Models" ***Proceedings of the 10th Annual DAMA-NCR Symposium***, Washington, DC September 11-12, 1997, pp. 103-160 (Proceedings available from DAMA - National Capital Region - 6729 Curran Street - McLean, VA 22101).

23 See the IHRIS Project Website at http://www.isy.vcu.edu/~paiken/dre/ihris.htm.

24 B. Inmon ***Data Architecture: The Information Paradigm*** QED Technical Publishing Group, 1993.

25 E. Chikofsky "The Necessity Of Data Reverse Engineering" in P. Aiken ***Data Reverse Engineering: Slaying the Legacy Dragon*** McGraw-Hill 1996, pp. 8-11.

26 F. Neiderman, J. Brancheau, J. Wetherbe "Information Systems Management Issues of the '90s" ***MIS Quarterly*** December 1991 15(4):475-502.

27 B. Caldwell "Missteps, Miscues: Business reengineering failures have cost corporations billions, and spending is still on the rise" ***InformationWeek*** June 20, 1994, pp. 50-60.

28 PAiken, P. H., Ngwenyama, O. K., and Broome, L. ***Reverse Engineering New Systems for Smooth Implementation***. ***IEEE Software***. March/April 1999 **16**(2):36-43.

29 See http://www.microsoft.com/repository/start.htm for details.

30 C. Finkelstein & P. Aiken ***Data Warehouse Engineering with Data Quality Reengineering*** scheduled for October 1998 publication by McGraw-Hill (ISBN 0-07-913705-9).