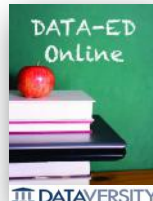


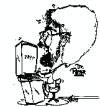
Data Modeling Fundamentals



peter.aiken@anythingawesome.com +1.804.382.5957



Achieving common understanding



© Copyright 2024 Peter Aiken, PhD Slide # 1

Peter Aiken, Ph.D.

- I've been doing this a long time
- My work is recognized as useful
- Associate Professor of IS (vcu.edu)
- Institute for Defense Analyses (ida.org)
- DAMA International (dama.org)
- MIT CDO Society (iscdo.org)
- Anything Awesome (anythingawesome.com)
- Experienced w/ 500+ data management practices worldwide
- 12 books and dozens of articles
- Multi-year immersions
 - US DoD (DISA/Army/Marines/DLA)
 - Nokia
 - Deutsche Bank **\$1,500,000,000.00** USD
 - Wells Fargo
 - Walmart
 - HUD ...



<http://anythingawesome.com>





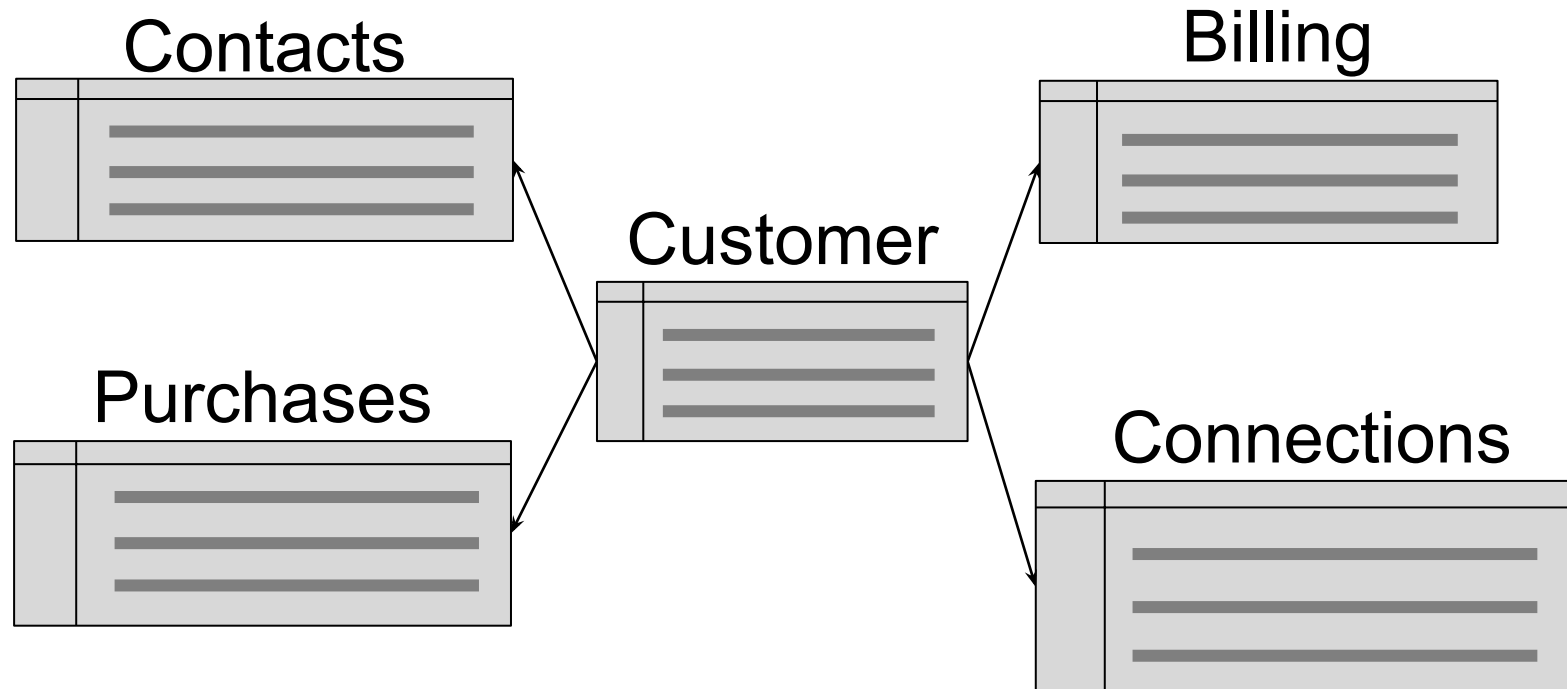
JSON Data Modeling

Matthew D. Groves, DevRel Engineer

FEBRUARY 2024

Modeling Data in a Relational World

RDBMS table-based modeling



Making a Change Using RDBMS can be complex

User Table

User ID	First	Last	Zip	Country ID
1	Frank	Wiegel	94040	001
2	Joe	Smith	94040	001
3	Ali	Dodson	94040	001
4	Sarah	Gorin	NW1	002
5	Bob	Young	30303	001
6	Nancy	Baker	10010	001
7	Ray	Jones	31311	001
8	Lee	Chen	V5V3M	008
...				...
50000	Doug	Moore	04252	001
50001	Mary	White	SW195	002
50002	Lisa	Clark	12425	001

Photo Table

User ID	Photo ID	Comment	Country ID
2	d043	NYC	001
2	b054	Bday	007
5	c036	Miami	001
7	d072	Sunset	133
5002	e086	Spain	133

Status Table

User ID	Status ID	Text	Country ID
1	a42	At conf	134
4	b26	excited	007
5	c32	hockey	008
12	d83	Go A's	001
5000	e34	sailing	005

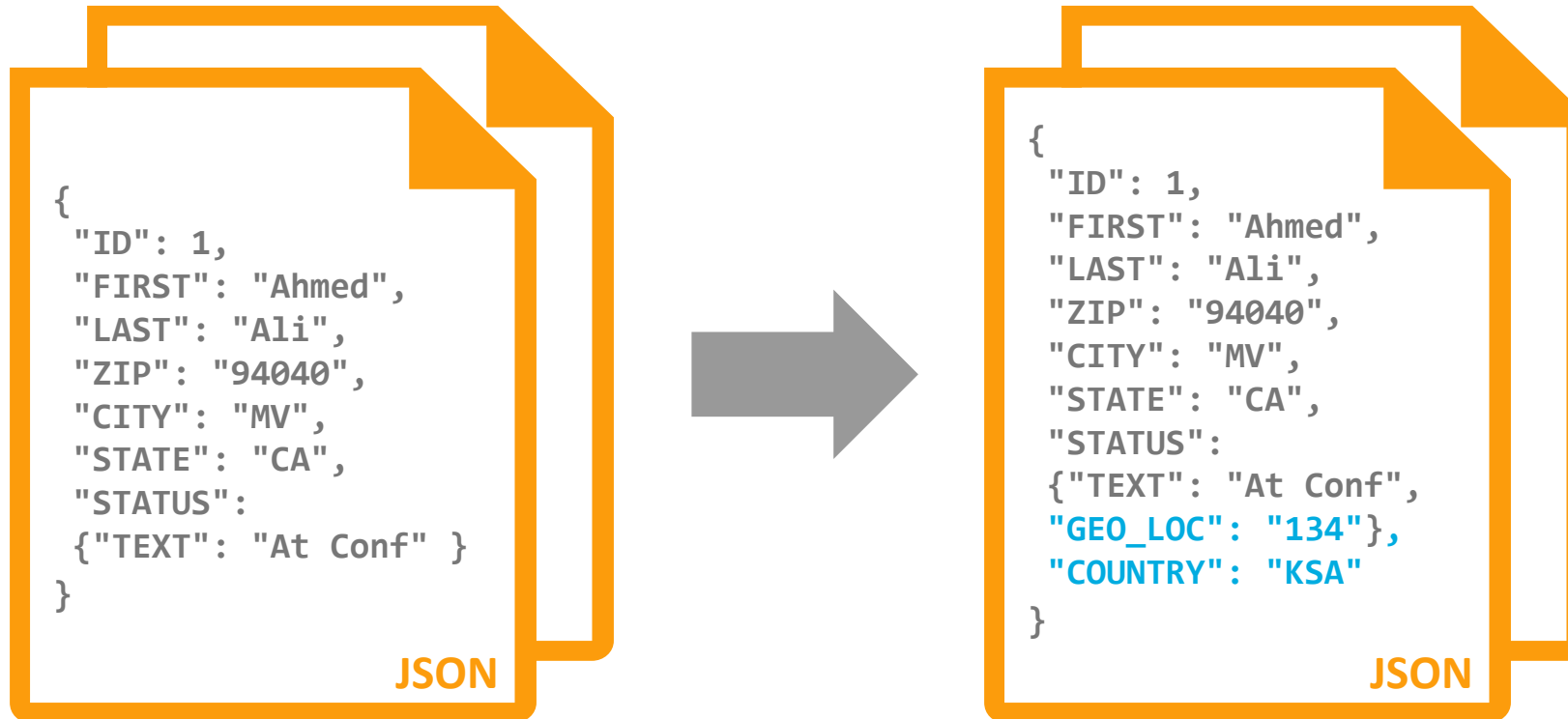
Affiliations Table

User ID	Affl ID	Affl Name	Country ID
2	a42	Cal	001
4	b96	USC	001
7	c14	UW	001
8	e22	Oxford	002

Country Table

Country ID	Country name
001	USA
002	UK
003	Argentina
004	Australia
005	Aruba
006	Austria
007	Brazil
008	Canada
009	Chile
...	
130	Portugal
131	Romania
132	Russia
133	Spain
134	Sweden

Making the Same Change With a Document Database



Just add information to each document

Embed vs Refer | General Considerations



EMBED WHEN

-
- There is an Ownership Relationship
 - Both docs are frequently accessed together
 - Reads greatly outnumber writes
 - Data is small

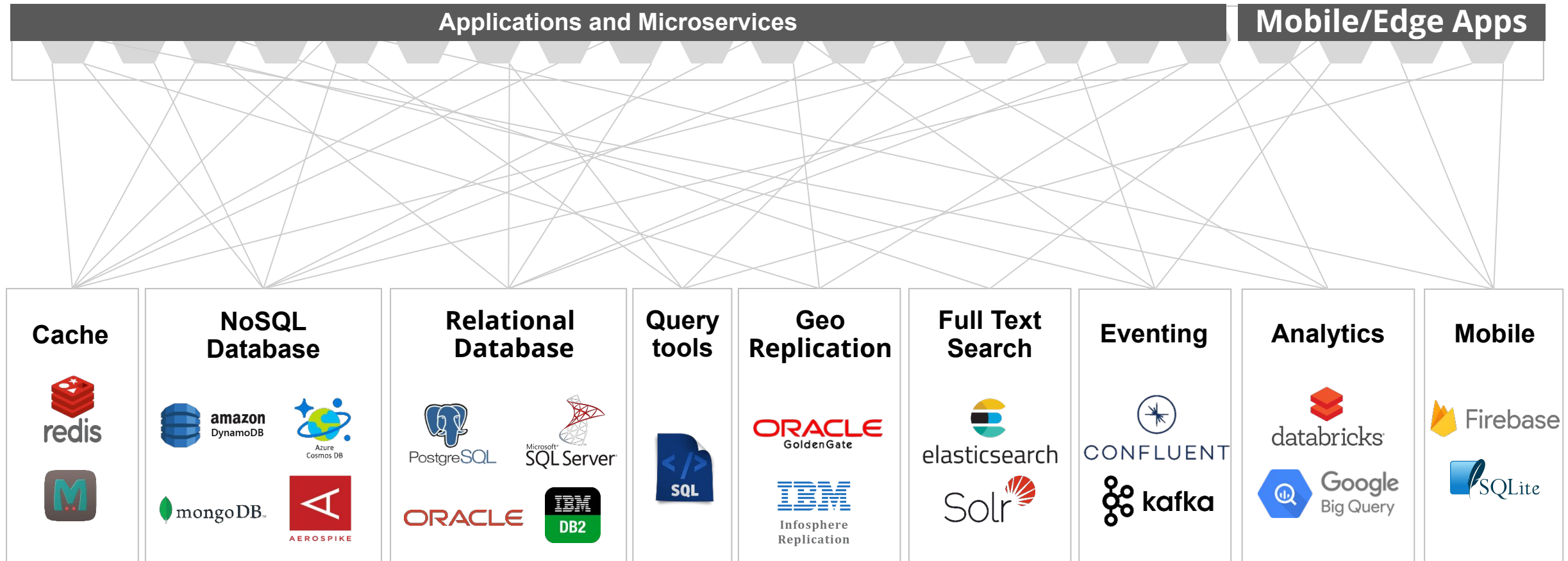


REFER WHEN

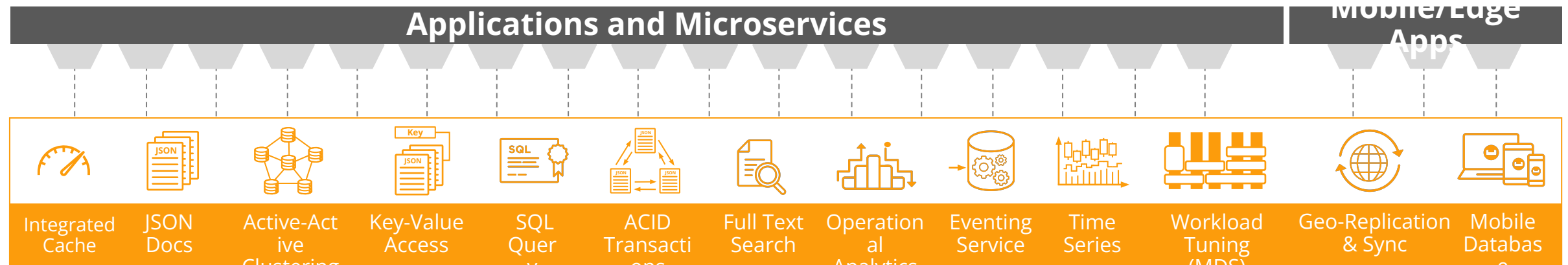
-
- There is not an Ownership Relationship
 - Both docs are not frequently accessed together
 - Document is updated frequently
 - Need to reduce the document size

Try to embed first, refer when it makes sense

Data Sprawl & Management Challenges



Couchbase Capella's Range of Capabilities



Fast

- Memory-first design
- Active-Active Clustering
- Geo-replication & sync
- Ultra-low latency

Affordable

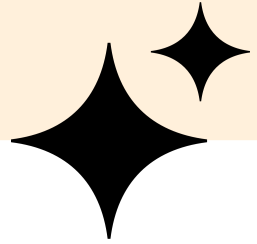
- Incredible price/performance
- Elastic scaling up & down
- High-density storage
- MDS resource scaling

Versatile

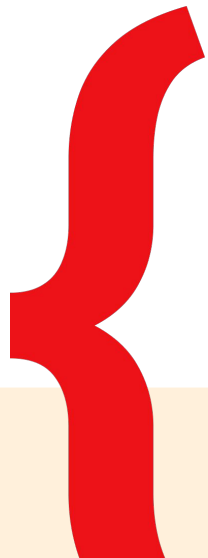
- Flexible JSON documents
- 7 Multimodel access services
- Composable features
- Mobile & Edge
- Peer-to-Peer Mobile Sync

Easy as SQL

- SQL++ is SQL for JSON
- Schema on-demand
- Multi-doc ACID SQL transactions
- HA, DR & backup service
- SDKs for 12+ languages



Thank you!

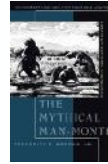


Couchbase



Data Representation is the Essence of Programming

- Mythical Man Month
 - 9 parallel effort x 1 month each \neq baby
- Fred Brooks Jr.'s observation
 - Data representation is the essence of programming
 - **"Show me your flowchart and conceal your tables, and I shall continue to be mystified.**
 - **Show me your tables, and I won't usually need your flowchart; it'll be obvious."**



(1931-2022)

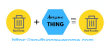
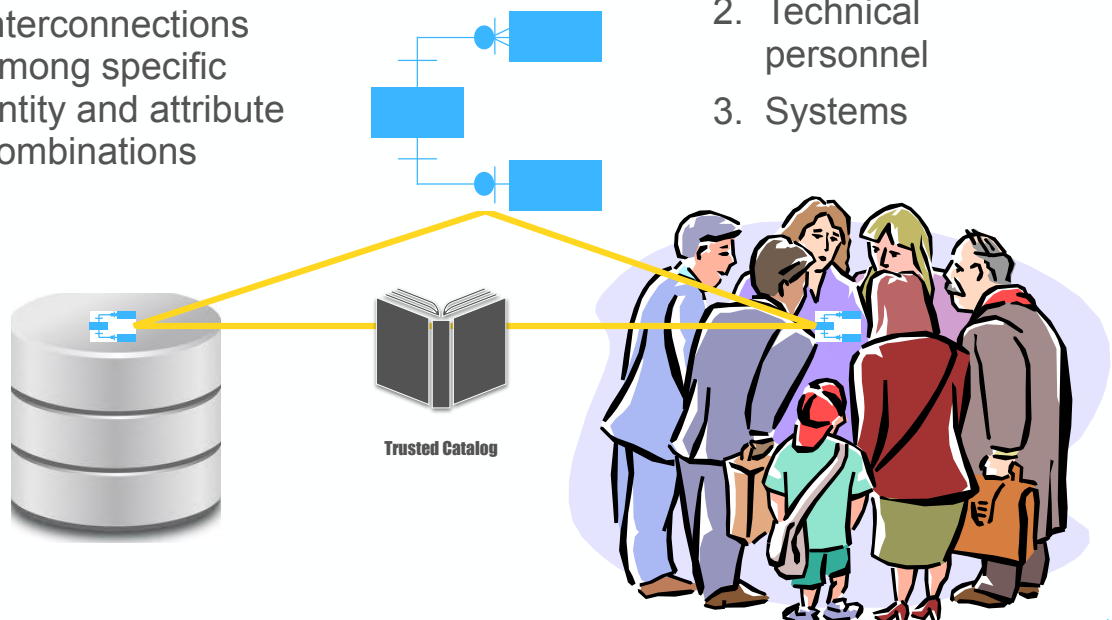


<https://creativecommons.org/licenses/by-nc-sa/4.0/>

© Copyright 2024 by Peter Allen Slide 3

Understanding a data model

- Documented and articulated as a digital blueprint of the commonalities and interconnections among specific entity and attribute combinations
- Shared by
 1. Business users
 2. Technical personnel
 3. Systems



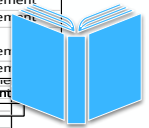
<https://creativecommons.org/licenses/by-nc-sa/4.0/>

© Copyright 2024 by Peter Allen Slide 4

Business Glossary

- Start of Enterprise Taxonomy
- Defines Initial Entities for Conceptual Data Model
- Engages the Business Community to Validate Entities and provide meaningful business definitions

Entity	Description	Domain Area
Donor	Funder	Business Development
Solicitations	Need for Work	Business Development
Solicitations Proposal	Response to Need for Work	Business Development
Pre-Positioning	Intelligence Gathering	Business Development
Award/Sub-Award	Funding Vehicle	Business Development
Terms Conditions	Details about a Funding Vehicle	Business Development
Budget	Amount of Money Available	Business Development
Work Plan	Set of Activities to Complete	Business Development
PMP	Monitoring Plan for Activities	Business Development
Project	An NGO Project is defined as a self-contained set of interventions or activities with the following characteristics: a) an external client; b) purchase order, contract or agreement; c) expected deliverables, outcomes and results; d) a beginning and end date of implementation; e) an approved budget; and full and/or part time NGO staff	Project Management
Geographic Area	Location in which a Central Office resides	Project Management
Office Locations		Project Management
Project Roles		Project Management
Project Artifacts		Project Management
Project Budget		Project Management
Project Work Plan		Project Management
Milestones	Schedule of completed activities	Project Management
Monitoring	Plan to measure Activities	Project Management
Evaluation	Assessment of Activities	Project Management
Indicators	Target of Outcome	Project Management
Outcomes	Statement of what needs to be accomplished	Project Management
Acct Receivable	Payments to NGO	Financial Management
Chart of Accounts	Defined Accounts	Financial Management
Payroll	Process to Pay Worker	Financial Management
Supplier	Provider of Goods or Service	Financial Management
Contract	Binding Agreement	Financial Management
Purchase Order	Statement of Good or Service	Financial Management
Performance	Level of Success	Talent Management
Benefits		Talent Management
Skills		Talent Management
Worker	Person who has been hired by NGO	Talent Management
Candidate	Potential hire of NGO	Talent Management



Trusted Catalog
© Copyright 2024 by Peter Allen Slide 5

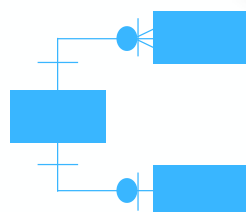
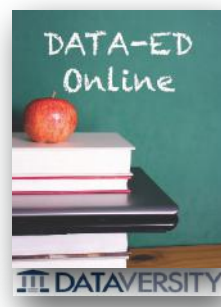


Program Overview

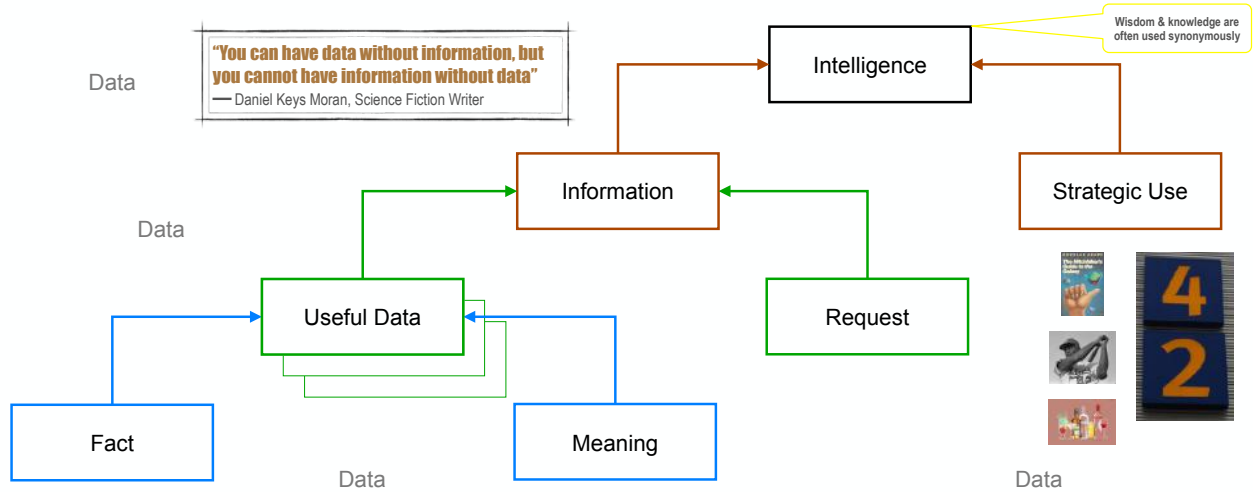
- What is data modeling required for?
 - Increasing understanding—systems to humans
 - Precisely defining data
 - Achieving simplification goals
 - Focused points of agreement
 - Understanding, building & deploying stable business models/strategy
- Why is modeling required for data understanding?
 - Why model anything?
 - Data requires precise definitions/agreements/understandings
 - Suboptimal data practices accumulate data debt
 - Data modeling describes important details about data that must be perfect
 - Should be considered from a primarily iterative development context
 - Understand and document data structures
- How using data models effectively
 - Leverages a standard communication form
 - Reduces corrosive anomalies
 - Keeps focus on motivation using purpose statements
 - Captures and communicates vital business information
 - Forward engineering (Goal=Development/Building)
 - Reverse engineering (Goal=Understanding)
- Take Aways, References, Q&A



Data Modeling Fundamentals
Achieving common understanding



A model distinguishing among 3 important concepts



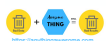
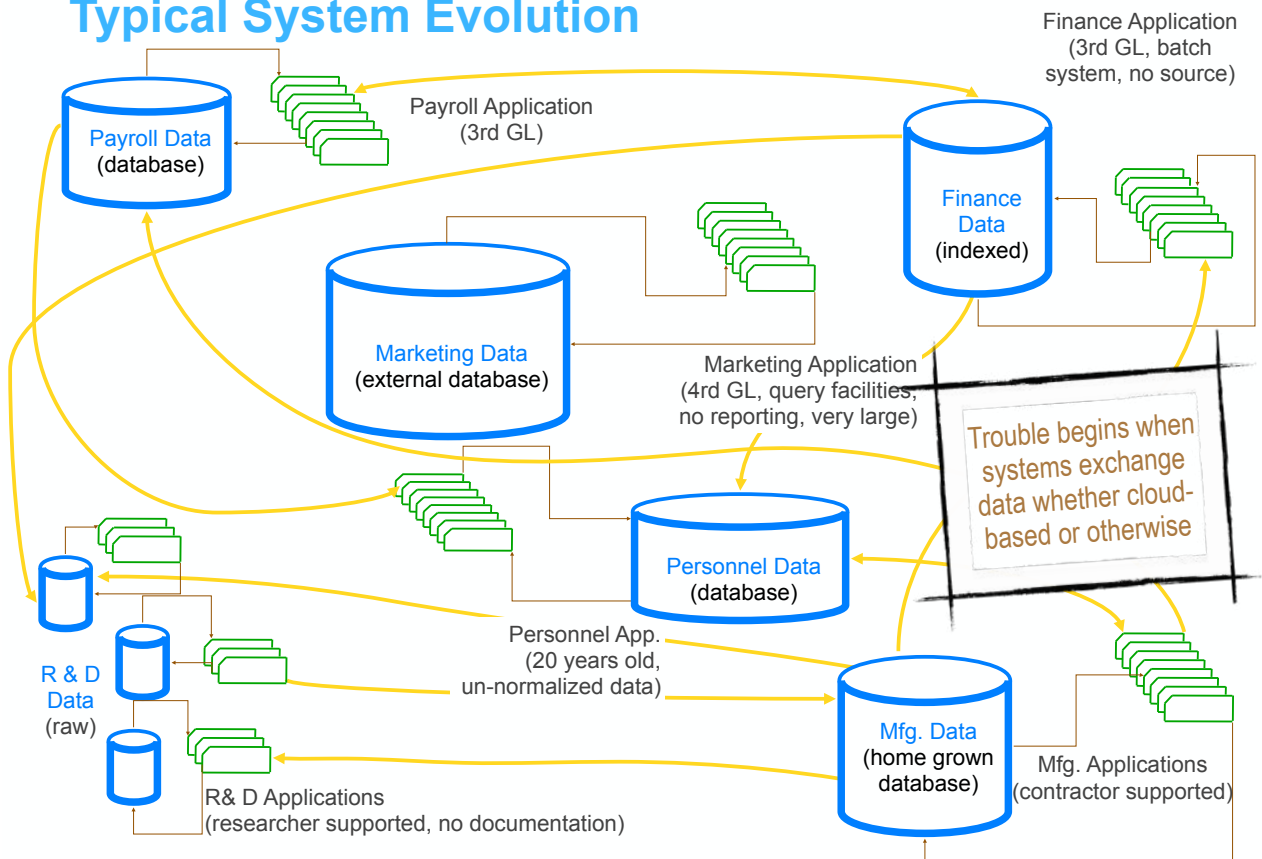
A copy of this bit is on YouTube: <https://youtu.be/seiKEXul7RI>

1. Each FACT combines with one or more MEANINGS.
2. Each specific FACT and MEANING combination is referred to as a DATUM.
3. An INFORMATION is one or more DATA that are returned in response to a specific REQUEST
4. INFORMATION REUSE is enabled when one FACT is combined with more than one MEANING.
5. INTELLIGENCE is INFORMATION associated with its STRATEGIC USES.
6. DATA/INFORMATION must formally arranged into an ARCHITECTURE.



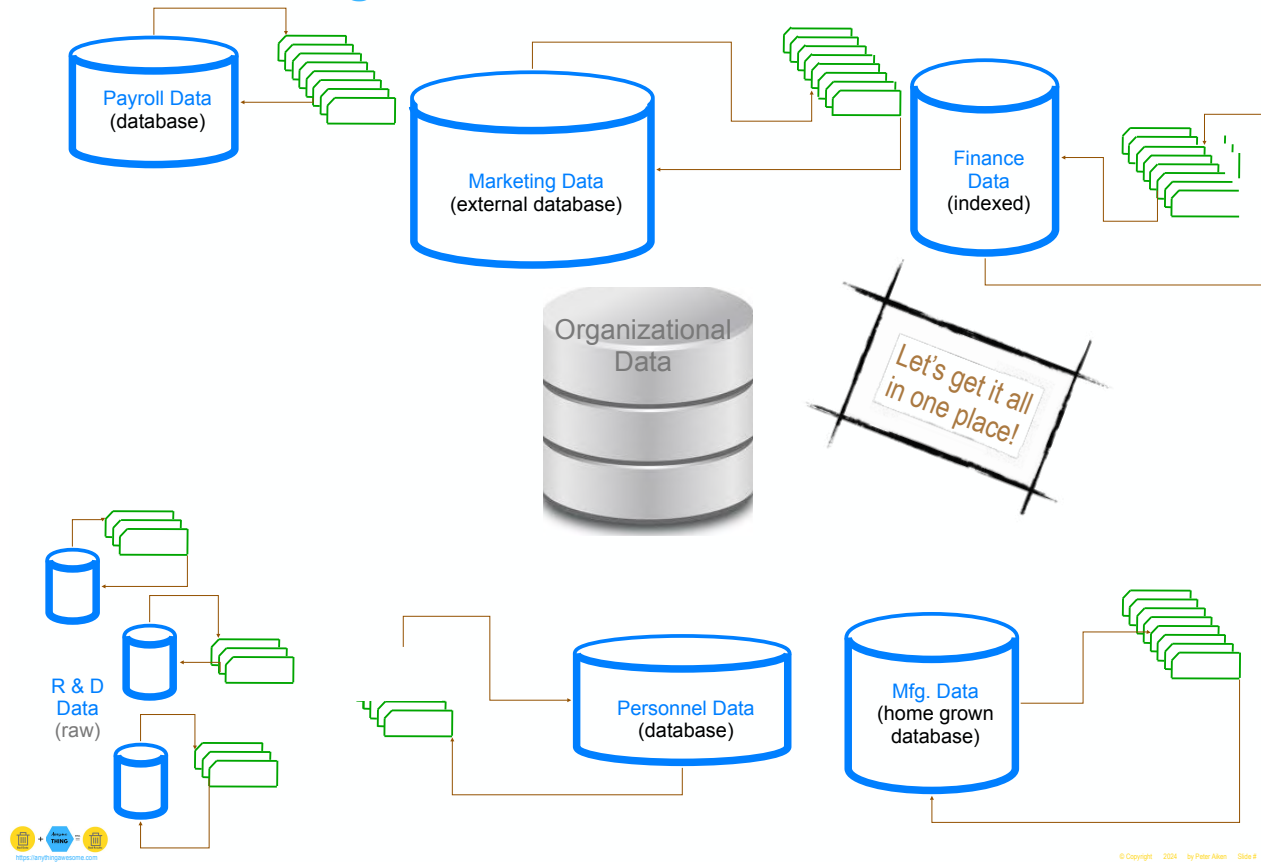
[Built on definitions from Dan Appleton 1983]
© Copyright 2024 by Peter Allen. Slide 7

Typical System Evolution

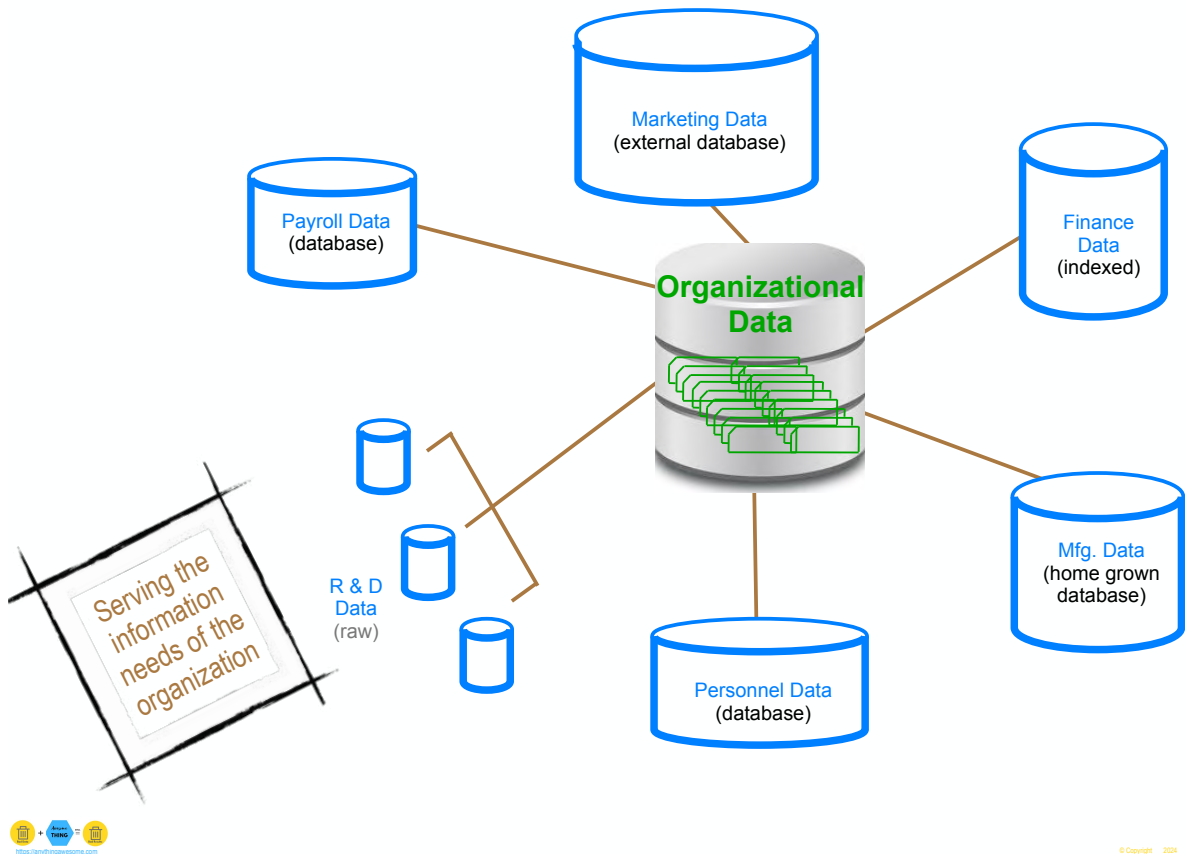


© Copyright 2024 by Peter Allen. Slide 8

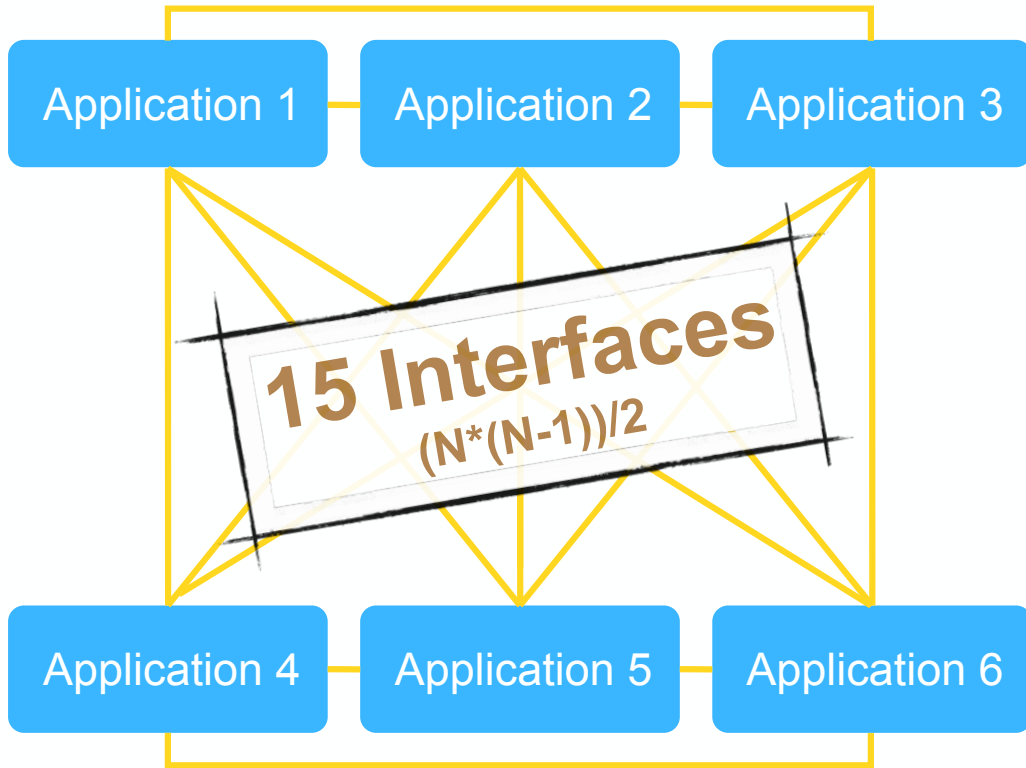
... Then Integrate



... Then Re-architect



How many interfaces are required to solve this integration problem?

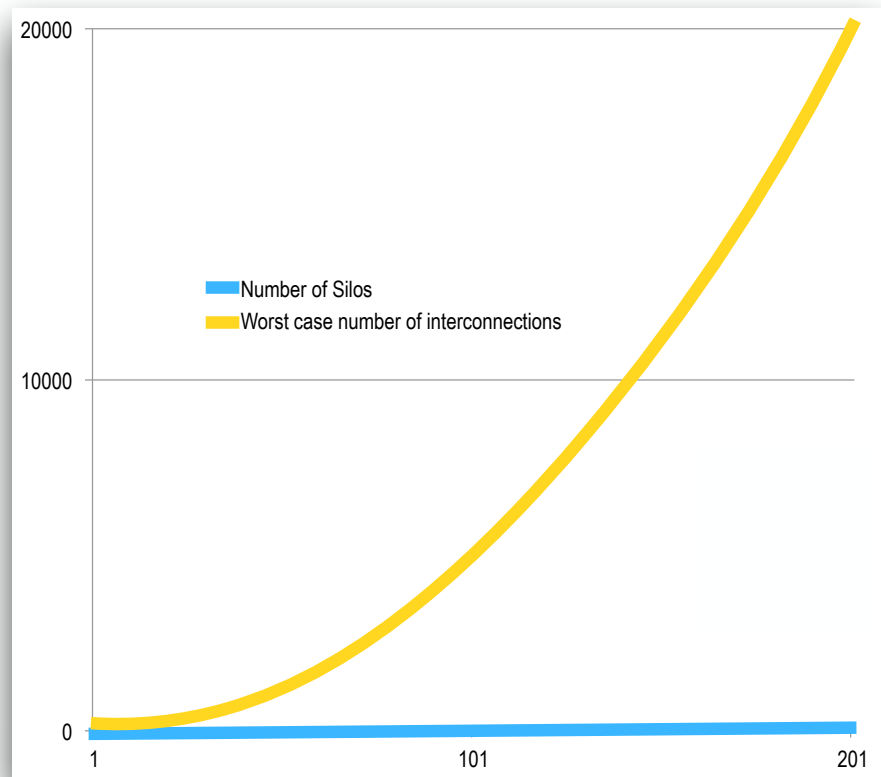


RBC: 200 applications - 4900 batch interfaces

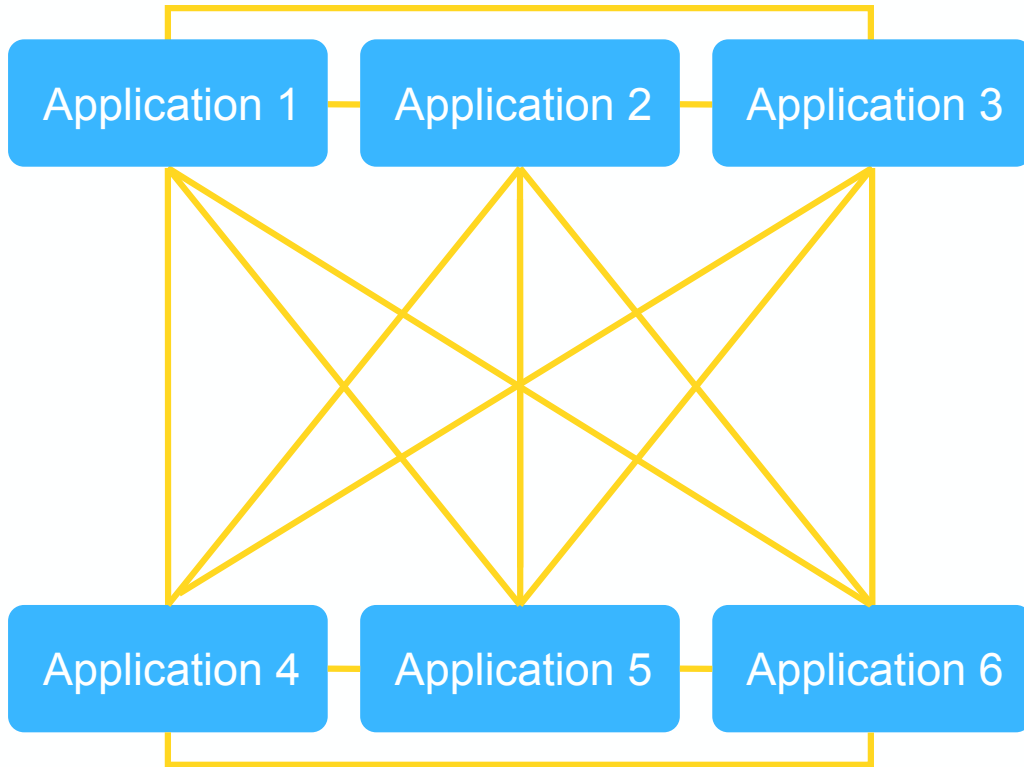


The rapidly increasing cost of complexity

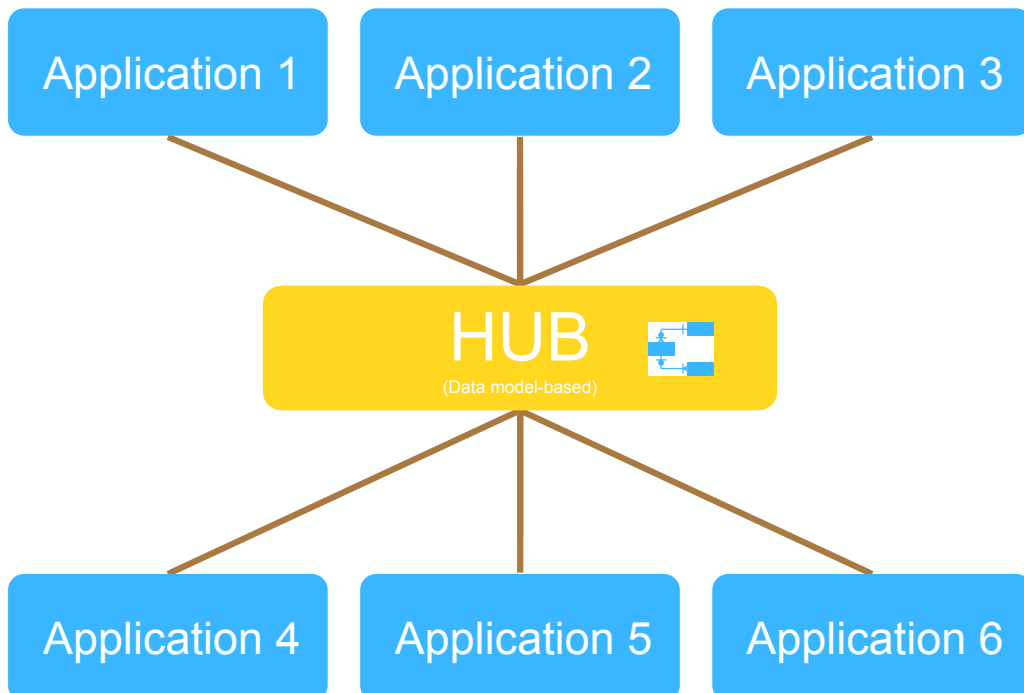
N
• 6 / 15
• 60 / 1,770
• 600 / 179,700
• 200 / 19,900
• 200 / 5,000 (actual)



Point to Point Integration

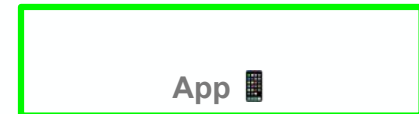
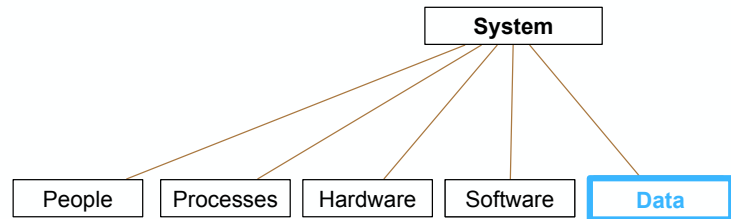
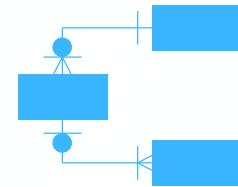


Spoke and Hub Integration



Data models ...

- Capture and maintain formal (usually physical) system data requirements
 - *An organized, purposeful structure regarded as a whole and consisting of interrelated/interdependent elements*
<http://www.businessdictionary.com/definition/system.html#ixzz2317LyAjI>
- Represent the lowest level of decomposition available in systems
 - People
 - Processes
 - Hardware
 - Software
 - Data
- Are by necessity, the most stable system component over time
- Incorporate organizational business rules
 - Can a project be owned by more than one department?

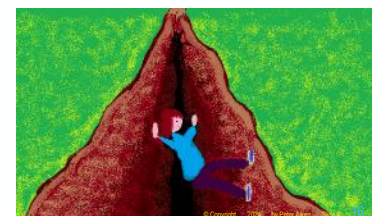
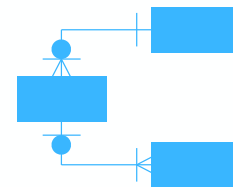
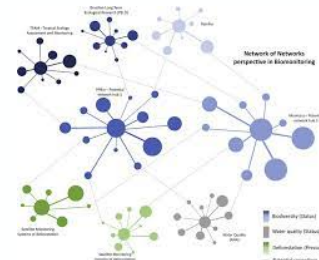


If it isn't correct at the data model representation, all other interpretations must be suspect

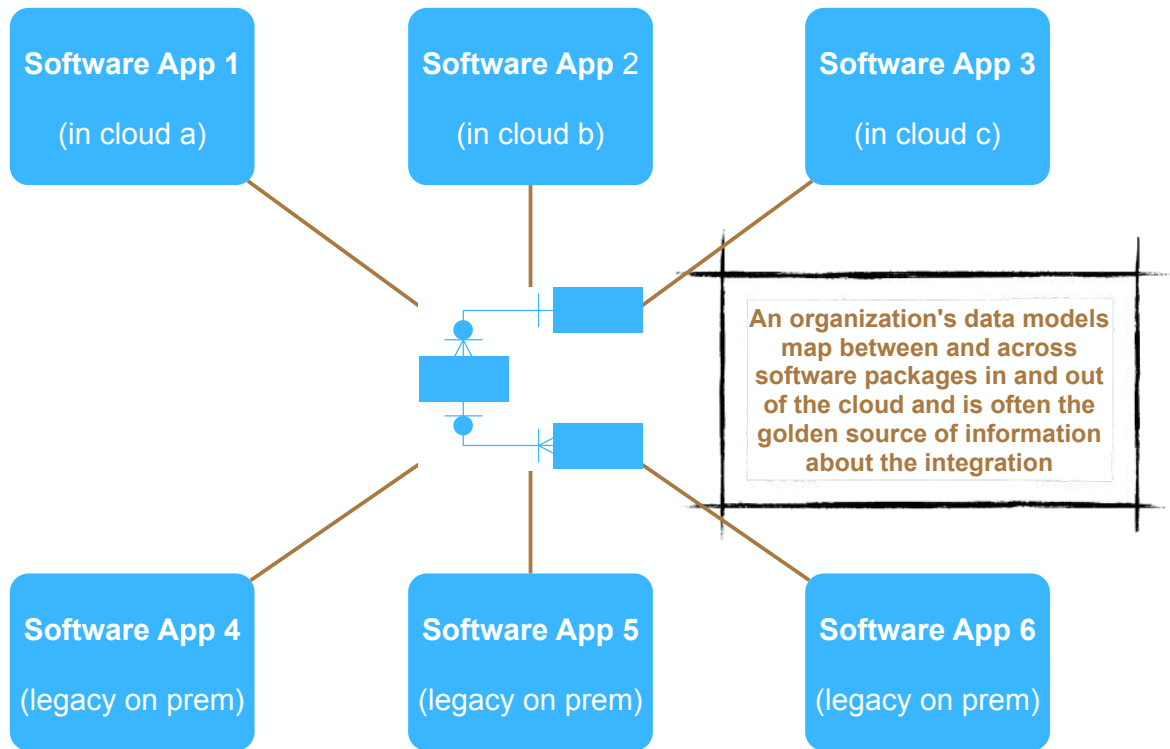


Minimize Interconnections

- As few HUBS as possible explicitly balancing various aspects of risk
 - Single point of failure
 - Easier to steal
 - Easier to manage
- A data model is at the heart of every HUB
 - Only reliable means of conveying the enormous amount of information required to run HUB for organizations
 - Also, objective use of standards within and across organizations
 - Can always be inferred or determined objectively
- Data modeling is not considered a necessary IT skill
 - Business and IT decision makers are unknowledgeable about data decisions
 - IT assigns responsibility for data to the business
 - Confusion has reigned for decades
 - Data debt has accumulated as a result

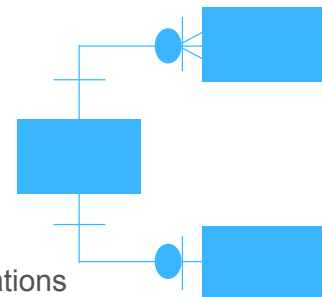


Today's IT environment



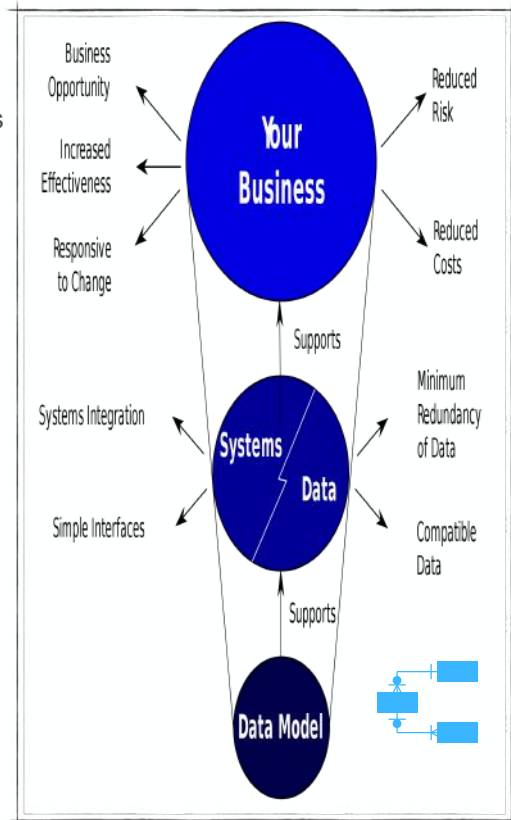
Definitions

- Model
 - A representation of something that exists or a pattern for something to be made
 - A model can contain one or more diagrams
 - Model diagrams using standard symbols that allow one to understand content
 - Maps, org charts, and building blueprints are examples of models in used daily
- Data Model
 - Analysis and design method used to define and analyze data requirements
 - Integrated collection of specifications and related diagrams that represent data requirements and design
 - Employs standardized text/symbols to represent data attributes (grouped into data elements) and the relationships among them
- Data Modeling
 - The process of discovering, analyzing, and scoping data requirements, and then
 - Representing and communicating the data requirements in a precise form called a *data model*.
 - Is used to design data structures that support the specifications



Stable (thus) Shareable Business & Data Models

- Represent stable sharable data
 - Data models are required to share data or information about data
 - Using mutually understood definitions
 - Candidate **enterprise standard** data
- Support stable business models
 - Data models are skeleton of the business architecture
 - They would form the 'bones' of a garden)
- Data organizations have remained the most stable business architecture component across my 40+ year career
- Are required as a condition of deployment
 - Cannot launch/ deploy without something
 - Many are not as organizationally useful as they could be (representing quantifiable opportunity costs)
 - Consider a **basic** part of any system documentation
 - Often missing

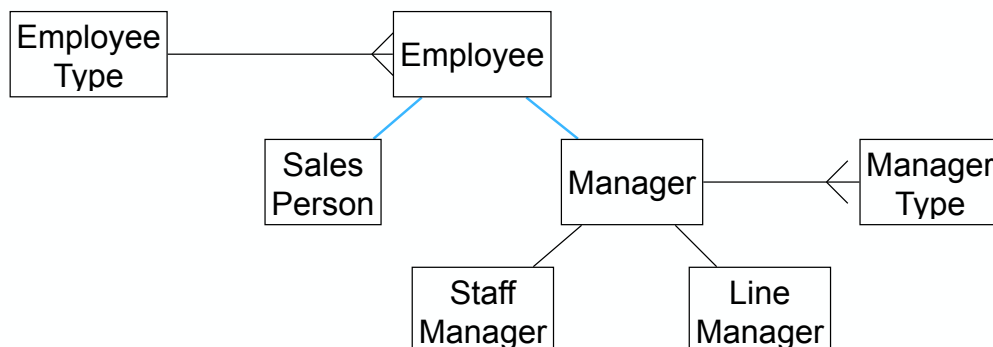
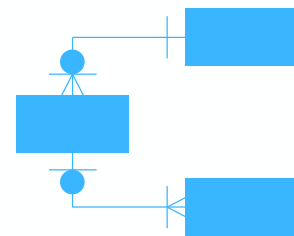


https://en.wikipedia.org/wiki/Data_model#/media/File:3-4_Data_model_roles.svg
 © Copyright 2004 by Peter Allen. Slide # 19



Data Models Used to Support Strategy

- Flexible, adaptable data structures
- Cleaner, less complex code
- Ensure strategy effectiveness measurement
- Build in future capabilities
- Form/assess merger and acquisitions strategies
 - For example: it was a poor design choice to require an employee to be either as salesperson or a manager (there is not structural support for managerial sales)
 - This decision 'haunted' the organization until the system was replaced



Program Overview

- What is data modeling required for?
 - Increasing understanding—systems to humans
 - Precisely defining data
 - Achieving simplification goals
 - Focused points of agreement
 - Understanding, building & deploying stable business models/strategy

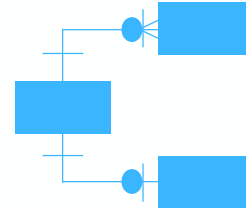


Data Modeling Fundamentals
Achieving common understanding

- Why is modeling required for data understanding?
 - Why model anything?
 - Data requires precise definitions/agreements/understandings
 - Suboptimal data practices accumulate data debt
 - Data modeling describes important details about data that must be perfect
 - Should be considered from a primarily iterative development context
 - Understand and document data structures



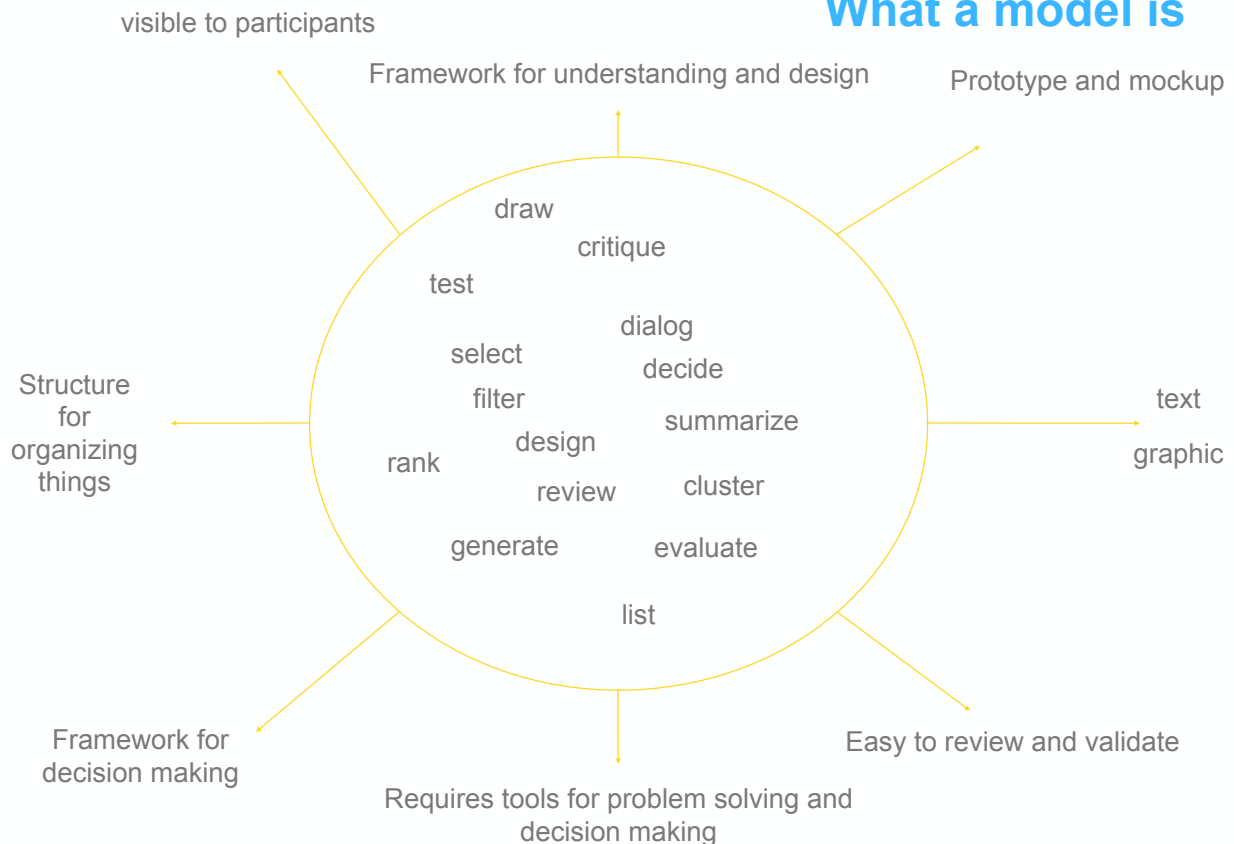
- How using data models effectively
 - Leverages a standard communication form
 - Reduces corrosive anomalies
 - Keeps focus on motivation using purpose statements
 - Captures and communicates vital business information
 - Forward engineering (Goal=Development/Building)
 - Reverse engineering (Goal=Understanding)



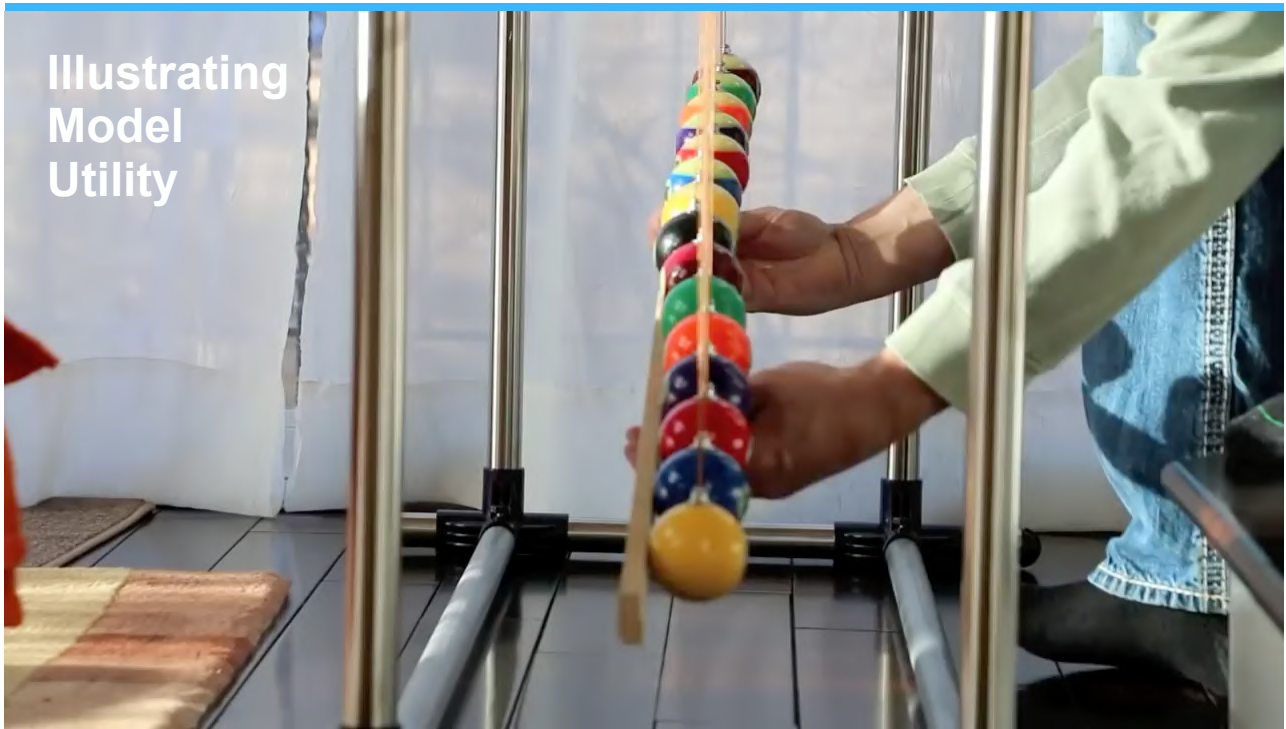
- Take Aways, References, Q&A



What a model is



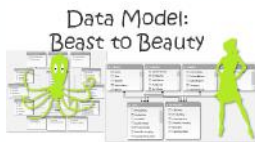
Illustrating Model Utility



- Represent expertise
- Store and formalize information
- Filter out extraneous detail
- Define an essential set of information
- Help understand complex system behavior
- Monitor/predict system responses to changing environmental conditions
- Communicate more effectively/efficiently
 - Novice/expert
 - Business/technical
 - Human/software
- Streamline documentation
- Monitor and predict system responses to changing conditions
- Gain information from the process of developing and interacting with the model
- Evaluate various scenarios or other outcomes indicated by the model
- Understand behaviors
- Illustrating patterns and metapatterns

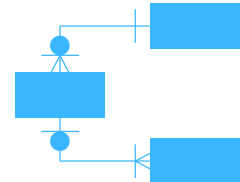


<https://www.youtube.com/watch?v=JslgubUJTck>

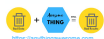


powerpivotpro.com

Why Model?



• Would you build a house without an architecture sketch?	• Model is the sketch of the system to be built in a project.
• Would you like to have an estimate how much your new house is going to cost?	• Your model gives you a very good idea of how demanding the implementation work is going to be!
• If you hired a set of constructors from all over the world to build your house, would you like them to have a common language?	• Model is the common language for the project team.
• Would you like to verify the proposals of the construction team before the work gets started?	• Models can be reviewed before thousands of hours of implementation work will be done.
• If it was a great house, would you like to build something rather similar again, in another place?	• It is possible to implement the system to various platforms using the same model.
• Would you drill into a wall of your house without a map of the plumbing and electric lines?	• Models document the system built in a project. This makes life easier for the support and maintenance!



Data Models Exist (whether you like it or not)

- All system have **data** models
 - Some are better **understood** and **documented** (and therefore more **useful** to the organization) than others



deviantart.com



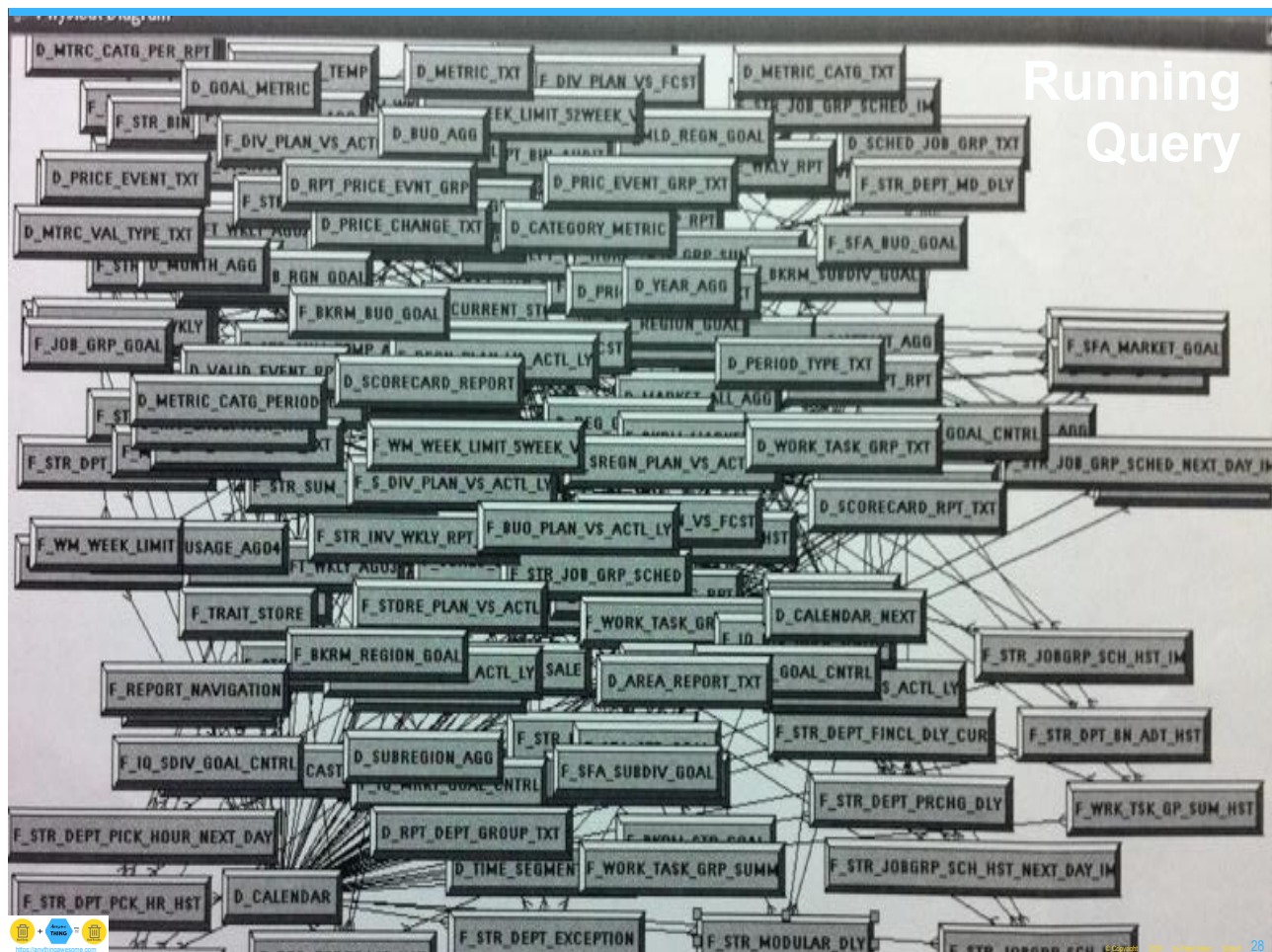
© Copyright 2024 by Peter Allen Slide # 25

Doing a poor job with data

- Failure to understand the role of data re: proposed and existing software/services
 - Locks in imperfections for the life of the application
 - Restricts data investment benefits
 - Decreases organizational data leverage
- Accounts for 20-40% of IT budgets devoted to evolving
 - Data **migration** (Changing the data location)
 - Data **conversion** (Changing data form, state, or product)
 - Data **improving** (Inspecting and manipulating, or re-keying data to prepare it for subsequent use)
- Lack of data capabilities causes everything else to
 - Take longer
 - Cost more
 - Deliver less
 - Present greater risk (with thanks to Tom DeMarco)



© Copyright 2024 by Peter Allen Slide # 26



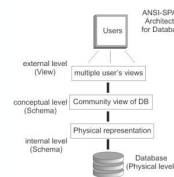
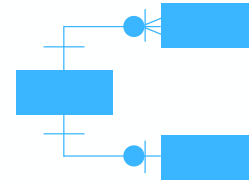
DEATH BY A THOUSAND CUTS



**unnecessary
discomfort
\$1.5 billion USD
from lots of
small cuts**

Data modeling

- The process of discovering, analyzing, and scoping data requirements
 - Understand what the data things are?
 - What do they do?
 - How do they interact?
- Representing/communicating requirements in a precise form called a data model
 - Maps of critical business assets
 - Compose and contain metadata essential to data consumers
 - Function as a kind of sheet music language
 - Metadata is essential to other business functions (definitions for governance, lineage for analytics, etc.)
- The process is iterative and may include a conceptual, logical, and physical model



process of discovering, analyzing, and scoping data requirements

- For organizational persons
places
things whose information needs to be

created
read
updated
deleted
archived

- These are called Attributes
 - Attributes are characteristics of "things"
 - Creating unique versions of each class instance



process of discovering, analyzing, and scoping data requirements

- An organization might decide to characterize the parts of a THING as:
 - Attributes: **ID**, **description**, **status**, **sex.to.be.assigned**, reserve.reason
- Decisions to manage information about each specific attribute has direct consequences
 - A decision to use the above data attributes permits the organization to determine if it has **female** THINGS are available to be **reserved**
- Characteristics can be shared
 - All THINGS may have a **status**
 - Many THINGS can be assigned to **females**
- Characteristics may be required to be unique
 - **ID** permits identification every THING as distinct for every other THING
 - **Description** is likely to be unique for each THING

THING
Thing.Id #
Thing.Description
Thing.Status
Thing.Sex.To.Be.Assigned
Thing.Reserve.Reason

Attributes arranged into an entity named "thing" – the attribute *Thing.Id* is the means used to identify a unique occurrence of thing



representing/communicating requirements in a precise form called a data model

- Q: What is an Attribute?
 - A characteristic of an instance in a collection of business things about which we create, read, update, and delete information
 - In this example, the attribute "club id" tells us much about this data collection
- What does the existence of this attribute tell us?
 - Clubs need to be identified (#) separately from one another
 - Club-specific information is likely maintained
 - Some concept (organization) exists above the 'club level'
 - ...



Attribute Definition



Club

Club ID #

current promotion

Maximum period of obligation

number cancelled ytd

number of members

total units sold for club

...

- Attributes describe an entity
- Attribute values are characteristics of “instances of business things”



Relationships specify entity connections

- Entities organized into a poorly designed model
- What is a Relationship?
 - Natural associations between two or more entities

Club

Club ID #

current promotion

Maximum period of obligation

number cancelled ytd

number of members

total units sold for club

...

ClubMember

ClubMember ID #

Club member first name

Club member last name

Club member net worth

...

ClusterConnector

ClubMember ID #

Club ID #

...



At Least 4 Variants of Modeling Notations

Chen Style

Ordinally - describes the minimum (optional vs mandatory) → M:N ← Cardinality - describes the maximum

1:N (n=0,1,2,3,...)
one to zero or more

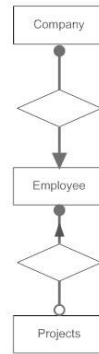
M:N (m and n=0,1,2,3,...)
zero or more to zero or more (many to many)

1:1
one to one



Bachman Style

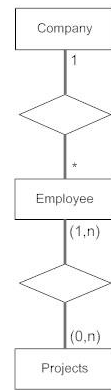
- one to one
- zero or more to one or more
- one to one or more



Most pick this

Martin Style

- 1 - one, and only one (mandatory)
- * - many (zero or more - optional)
- 1...* - one or more (mandatory)
- 0...1 - zero or one (optional)
- (0,1) - zero or one (optional)
- (1,n) - one or more (mandatory)
- (0,n) - zero or more (optional)
- (1,1) - one and only one (mandatory)



Information Engineering

- Exactly One (mandatory)
- One or Many (mandatory)
- Eventually One (optional)
- Zero, or Many (optional)
- Eventually One or Many (optional)



Possible Entity Relationship Cardinality Options

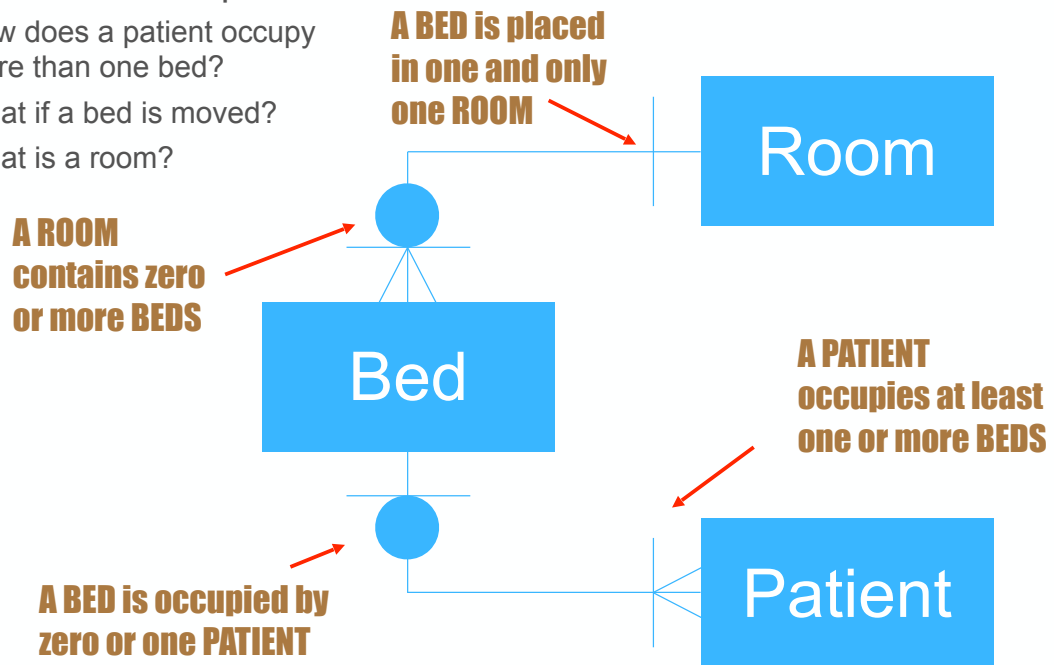
(For Information Engineering)

- Exactly One (mandatory)
- One or Many (mandatory)
- Eventually One (optional)
- Zero, or Many (optional)
- Eventually One or Many (optional)

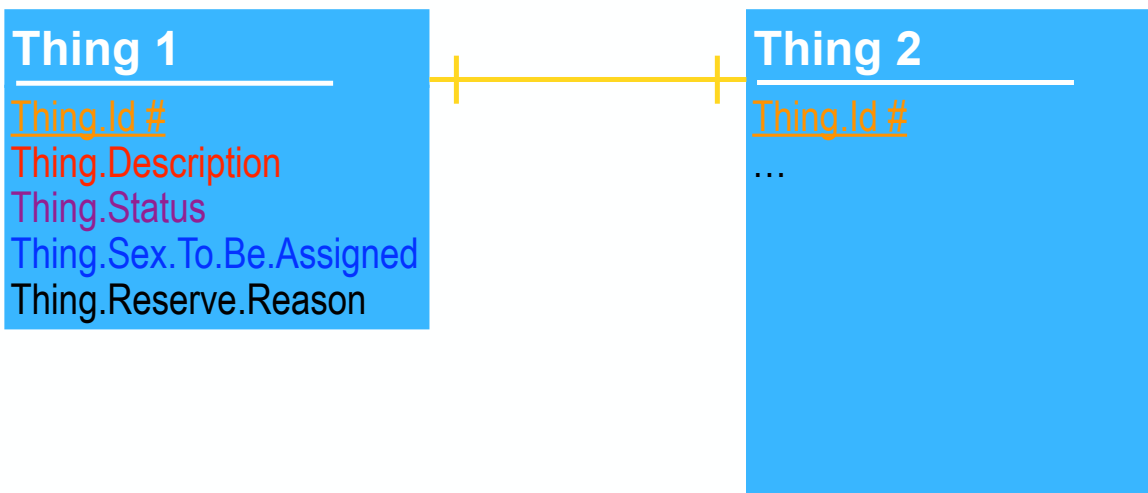


Ordinality and Cardinality Refine Relationships

- Defines mandatory/optional relationships using minimum/maximum occurrences from one entity to another
- Flaws in this example
 - How does a patient occupy more than one bed?
 - What if a bed is moved?
 - What is a room?



representing and communicating these in a precise form called a data model



Each THING 2 must be accompanied by a THING 1



Data Maps at the Entity Level → Stored Facts



a BED is related to a ROOM



More precision:
many BEDS are related to many ROOMS

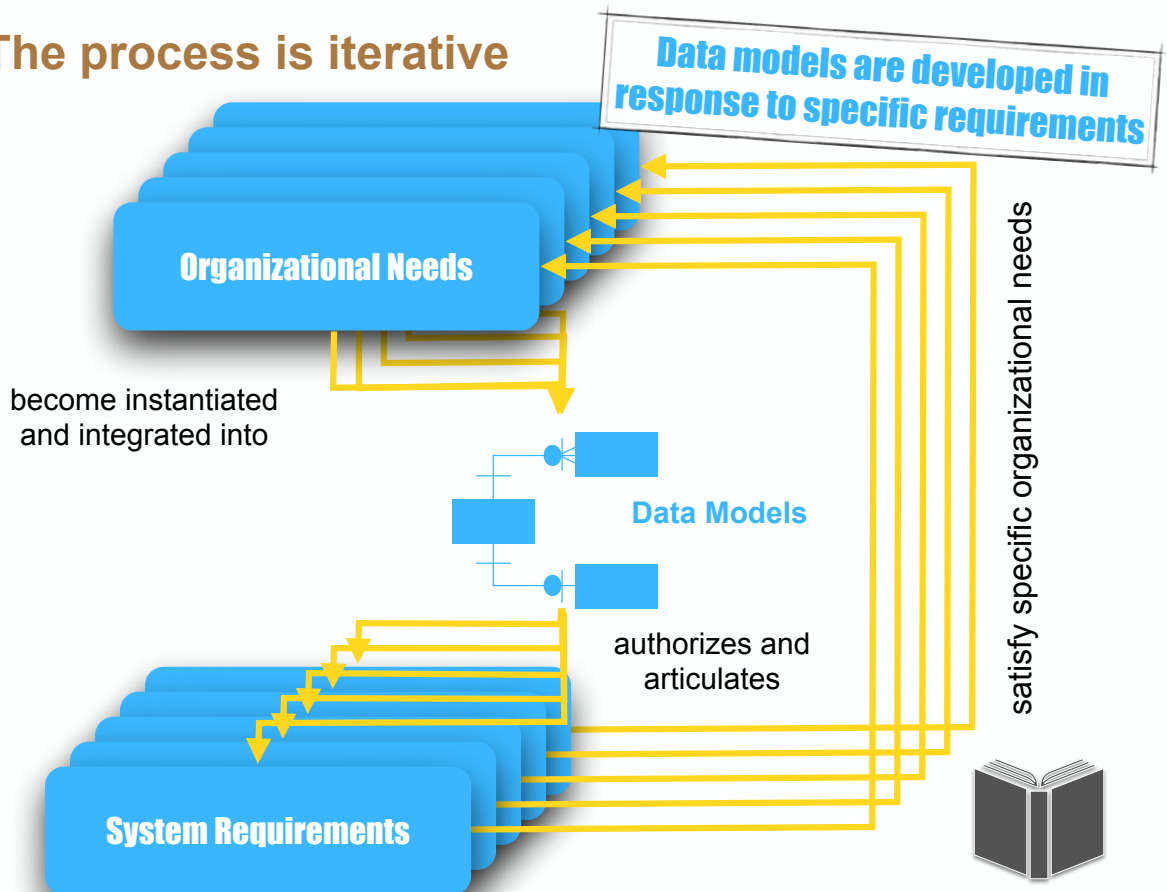
What if beds can be moved?



Better information:
many BEDS may be contained in each ROOM and each room may contain many beds



The process is iterative



Each data arrangement is a data structure

Data structure characteristics

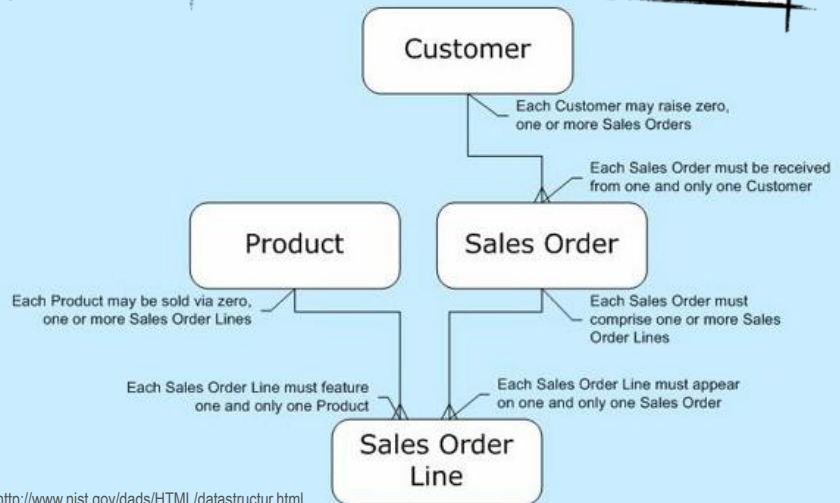
- Grammar for data objects
 - Grammar is the principles or rules of an art, science, or technique "a grammar of the theater"

Constraints for data objects

- Uniqueness
- Order
 - Hierarchical
 - Relational
 - Sequential
 - Network
 - Lake
 - other
- Balance
- Optimality

"An organization of information, usually in memory, for better algorithm efficiency, such as queue, stack, linked list, heap, dictionary, and tree, or conceptual unity, such as the name and address of a person. It may include redundant information, such as length of the list or number of nodes in a subtree."

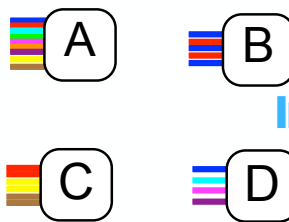
The fewer the better!



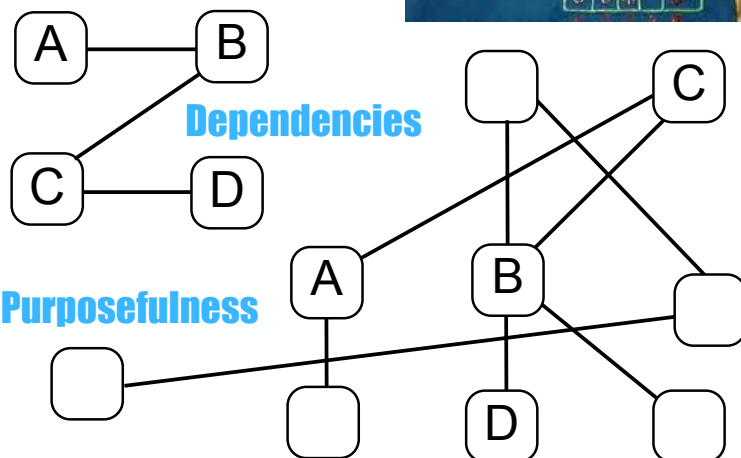
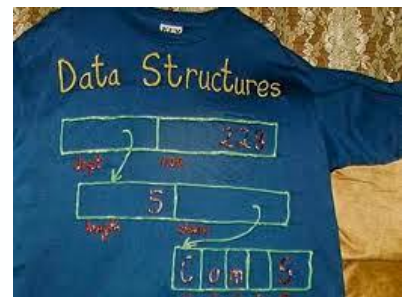
<http://www.nist.gov/dads/HTML/datastructur.html>

How are components expressed as data models?

- Details are organized into larger components
- Larger components are organized into data models
- Data models are organized into architectures



Intricate



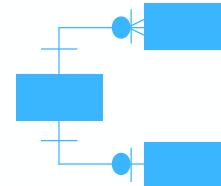
How are data models expressed as architectures?

- Attributes are organized into entities/objects
 - Attributes are characteristics of "things"
 - Entities/objects are "things" whose information is managed in support of strategy
 - Example(s)
- Entities/objects are organized into models
 - Combinations of attributes and entities are structured to represent information requirements
 - Poorly structured data, constrains organizational information delivery capabilities
 - Example(s)
- Models are organized into architectures
 - When building new systems, architectures are used to plan development
 - More often, the business cannot make use of them in support of strategy implementation because the architectures are not understood

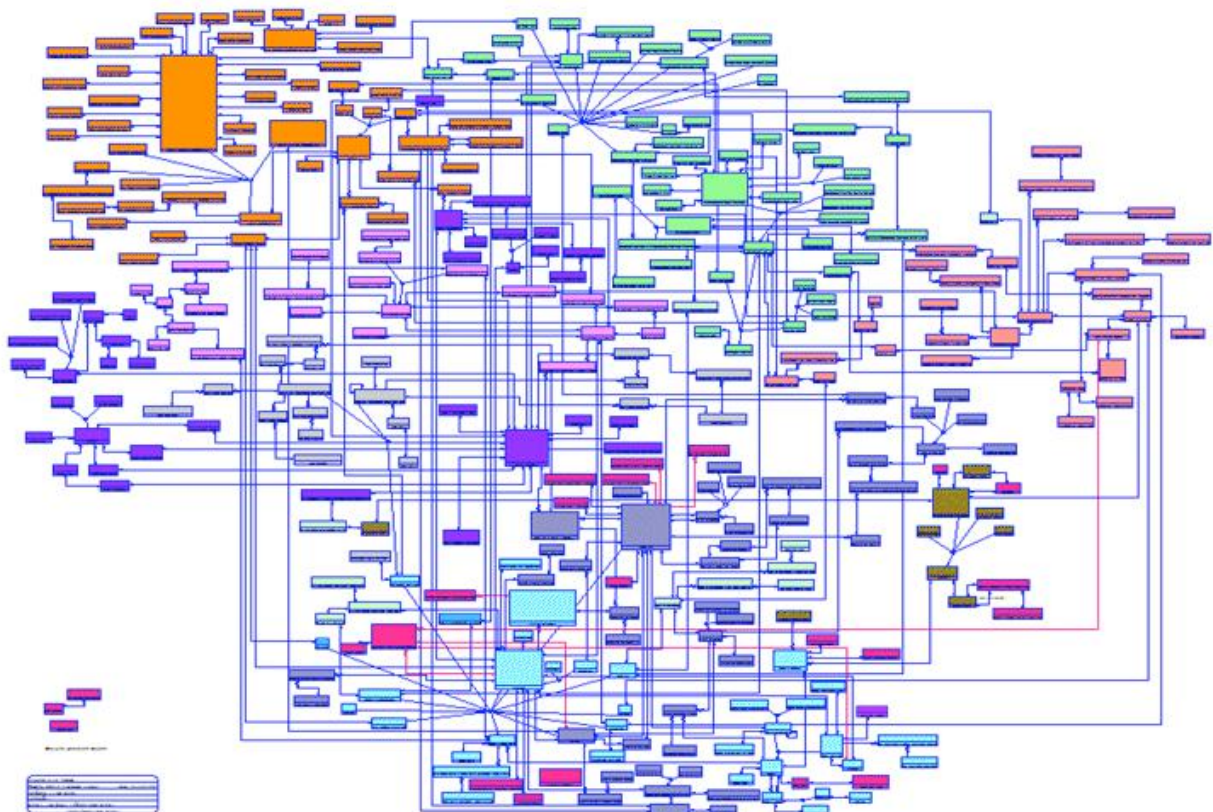
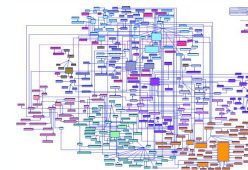
Intricate

THING
Thing.Id #
Thing.Description
Thing.Status
Thing.Sex.To.Be.Assigned
Thing.Reserve.Reason

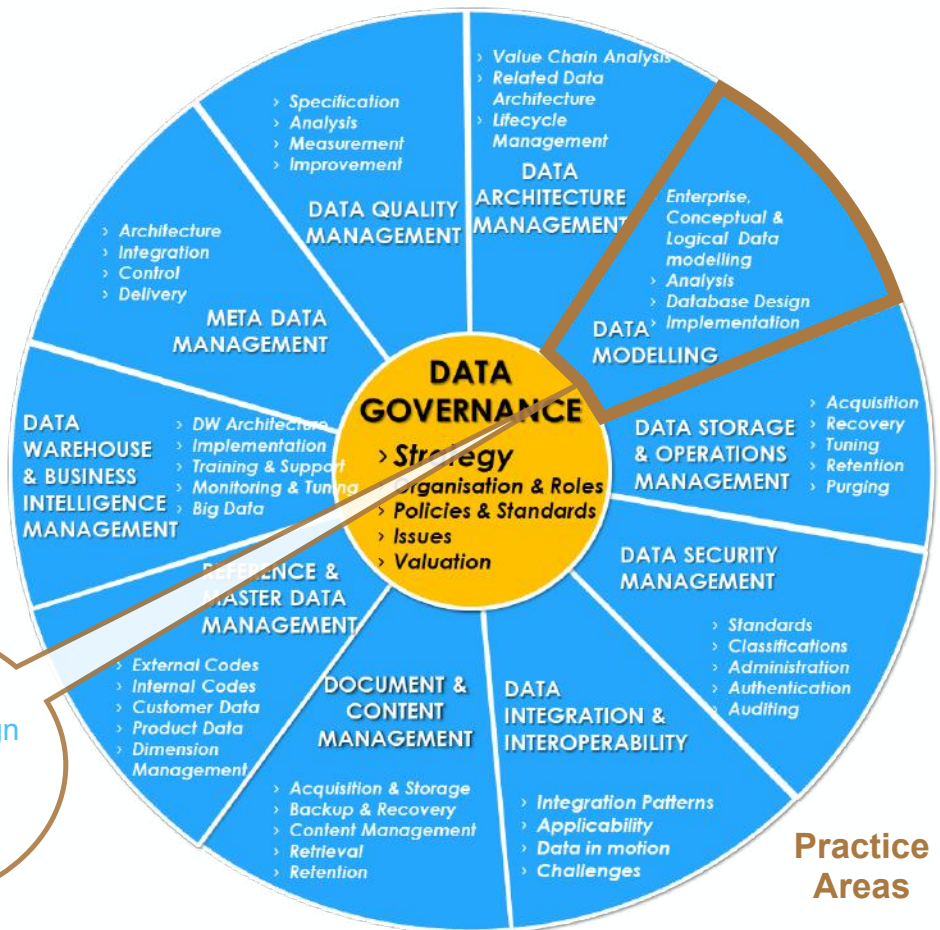
Dependencies



Purposefulness



- Analysis
- Database Design
- Implementation
- Additional data development



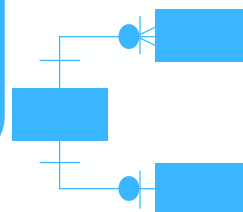
Practice Areas

Program Overview

- What is data modeling required for?
 - Increasing understanding—systems to humans
 - Precisely defining data
 - Achieving simplification goals
 - Focused points of agreement
 - Understanding, building & deploying stable business models/strategy
- Why is modeling required for data understanding?
 - Why model anything?
 - Data requires precise definitions/agreements/understandings
 - Suboptimal data practices accumulate data debt
 - Data modeling describes important details about data that must be perfect
 - Should be considered from a primarily iterative development context
 - Understand and document data structures
- How using data models effectively
 - Leverages a standard communication form
 - Reduces corrosive anomalies
 - Keeps focus on motivation using purpose statements
 - Captures and communicates vital business information
 - Forward engineering (Goal=Development/Building)
 - Reverse engineering (Goal=Understanding)
- Take Aways, References, Q&A



Data Modeling Fundamentals
Achieving common understanding



Where are your
(data) blueprints?

APPROVED

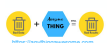


51

There are correct ways to organize data

- Optimization can be done for:
 - Flexibility
 - Adaptability
 - Retrievability
 - Risk reduction
 - ...
- Techniques include:
 - Data integrity
 - Smart codes bad/dumb codes good
 - Architecture (table joins)
 - ...

<u>Record</u>	<u>Purchaser ID</u>	<u>Song</u>	<u>Price</u>
1	Purchaser #1	Cool Walk (Live)	\$1.99
2	Purchaser #1	Sushi (Live)	\$1.29
3	Purchaser #1	Love Ballade (Live)	\$0.99
4	Purchaser #2	A Salute to Bach (Medley)	\$0.99



© Copyright 2024 by Peter Allen Slide 52

How Should it be Done? (Smart codes bad, dumb codes good)

- 804 → N zero N → long distance call signaling

- All telephone switching equipment (hardware) had to be changed to not route calls to long distance if the hardware 'saw' a zero in the middle of a 3 digit number



- Course listings

- "You can not add another undergraduate business computer course"

BUS3??

BUS360	
BUS361	
BUS362	
BUS363	
BUS364	
BUS365	
BUS366	
BUS367	
BUS368	
BUS369	

Business
Computer
Courses

- A large organization has to expand a primary master data item by a number of digits

- Requires upwards of 100,000 changes to be managed

https://www.youtube.com/watch?v=f1gwAGZs0&frag=pl%2Cwm



© Copyright 2004 by Peter Allen Slide 53

Table Handling

Record	Purchaser ID	Song	Price
1	Purchaser #1	Cool Walk (Live)	\$1.99
2	Purchaser #1	Sushi (Live)	\$1.29
3	Purchaser #1	Love Ballade (Live)	\$0.99
4	Purchaser #2	A Salute to Bach (Medley)	\$0.99

- A table is a collection of data items that have the same description, such as account totals or monthly averages; it consists of a table name and subordinate items called table elements.

- Under representation of other database characteristics causes confusion and introduces risk to organizational data capabilities

- In this example, the table consists of Song, Album

- and length?

- No, iTunes uses

- Song Start Time

- Song Stop Time

- More flexible and less risky



(a hypothetical portion of the) iTunes Music database

- Question
 - What information is lost if we delete record #1?

<u>Record</u>	<u>Purchaser ID</u>	<u>Song</u>	<u>Price</u>
1	Purchaser #1	Cool Walk (Live)	\$0.99
2	Purchaser #1	Sushi (Live)	\$1.29
3	Purchaser #1	Love Ballade (Live)	\$0.99
4	Purchaser #2	A Salute to Bach (Medley)	\$0.99



Music database: Deletion Anomaly

- Question
 - What information is lost if we delete record #1?
- Answer
 - We lose the fact that "Purchaser #1" purchased "Cool Walk (Live)"
 - We lose the fact that "Cool Walk (Live)" costs \$0.99
- This is usually undesirable and unintended

<u>Record</u>	<u>Purchaser ID</u>	<u>Song</u>	<u>Price</u>
1	Purchaser #1	Cool Walk (Live)	\$0.99
2	Purchaser #1	Sushi (Live)	\$1.29
3	Purchaser #1	Love Ballade (Live)	\$0.99
4	Purchaser #2	A Salute to Bach (Medley)	\$0.99



Music database: Insertion Anomalies

- Question:
 - Suppose we want to add new song "Cakewalk (Live)" and that it costs \$1.29?
 - This is **fact #1**
- Answer:
 - Cannot enter it until some future purchaser buys a copy of the song - **Fact #2**
 - We cannot insert a full row until we have an additional fact about that row
 - This is usually undesirable and unintended

Record	Purchaser ID	Song	Price
1	Purchaser #1	Cool Walk (Live)	\$1.99
2	Purchaser #1	Sushi (Live)	\$0.99
3	Purchaser #1	Love Ballade (Live)	\$0.99
4	Purchaser #2	A Salute to Bach (Medley)	\$0.99
5	Purchaser #?	Cakewalk (Live)	\$1.29

Fact #2 (under Purchaser #?) **Fact #1** (under Cakewalk (Live) and \$1.29)



Music database: Update Anomalies

- Question:
 - Suppose we want to change the price of "Cool Walk (Live)" from \$1.99 to \$1.29?
- Answer:
 - Change to data items such as Song requires examination of every single record
 - Will not catch spelling errors - such as "Coolwalk (Live)"
 - This is usually undesirable and unintended

Record	Purchaser ID	Song	Price
1	Purchaser #1	Cool Walk (Live)	\$1.99
2	Purchaser #1	Sushi (Live)	\$0.99
3	Purchaser #1	Love Ballade (Live)	\$0.99
4	Purchaser #2	A Salute to Bach (Medley)	\$0.99
5	Purchaser #3	Coolwalk (Live)	\$1.99



How Should it be Done? (In General)

- As much as possible, store 1 fact per row
 - Row 2 is a good example as it shows both that Purchaser #1 has purchased Sushi (Live) and that it costs \$0.99
 - These are two distinct facts and are correctly stored in two tables sharing a formal relationship
 - More remains codes

ORIGINAL

Record	Purchaser ID	Song	Pric
1	Purchaser #1	Cool Walk (Live)	\$1.99
2	Purchaser #1	Sushi (Live)	\$0.99
3	Purchaser #1	Love Ballade (Live)	\$0.99
4	Purchaser #2	A Salute to Bach	\$0.99
5	Purchaser #3	Coolwalk (Live)	\$1.99

PRICING

Record	Song	Price
1	Cool Walk (Live)	\$1.99
2	Sushi (Live)	\$0.99
3	Love Ballade (Live)	\$0.99
4	A Salute to Bach	\$0.99
5	Coolwalk (Live)	\$1.99

PURCHASES

Row	Purchaser ID	Song
1	Purchaser #1	Cool Walk (Live)
2	Purchaser #1	Sushi (Live)
3	Purchaser #1	Love Ballade (Live)
4	Purchaser #2	A Salute to Bach (Medley)
5	Purchaser #3	Coolwalk (Live)
6	Purchaser #3	A Salute to Bach (Medley)



How Should it be Done? (Joining Tables)

- Data from the two tables is *joined* to provide requested information
- The name variants for Cool Walk have been resolved
- PRICING table is a more flexible, adaptable engineered solution

(each **price** instance can provide context for many **purchase** instances)

PRICING

Record	Song	Price
1	Cool Walk (Live)	\$1.99
2	Sushi (Live)	\$0.99
3	Love Ballade (Live)	\$0.99
4	A Salute to Bach	\$0.99

PURCHASES

Row	Purchaser ID	Song
1	Purchaser #1	Cool Walk (Live)
2	Purchaser #1	Sushi (Live)
3	Purchaser #1	Love Ballade (Live)
4	Purchaser #2	A Salute to Bach (Medley)
5	Purchaser #3	Cool Walk (Live)
6	Purchaser #3	A Salute to Bach (Medley)





WIKIPEDIA
The Free Encyclopedia

- Main page
- Contents
- Featured content
- Current events
- Random article
- Donate to Wikipedia
- Wikipedia store

Interaction

- Help
- About Wikipedia
- Community portal
- Recent changes



Article [Talk](#)

Read [Edit](#) [View history](#)

Bed

Definition of Bed



From Wikipedia, the free encyclopedia

For other uses, see [Bed \(disambiguation\)](#).

A **bed** is a piece of **furniture** which is used as a place to **sleep** or **relax**.^{[1][2]}

Most modern beds consist of a soft, cushioned **mattress** on a **bed frame**, the mattress resting either on a solid **base**, often wood slats, or a sprung base.

Many beds include a **box spring inner-sprung base**, which is a large mattress-sized box containing wood and springs that provide additional support and



Bedroom on the [Detmold Open-air Museum](#) premises

Standard definitions might not provide sufficient context

The screenshot shows a software window with a title bar and a toolbar. The main area is divided into two sections. The top section has a header 'Name' and a sub-header 'Entity'. Below this, the word 'BED' is displayed in large, bold, black letters. The bottom section has a header 'Volumetrics' and a sub-header 'Definition'. Below this, there is a large text input field with the text 'Something you sleep in'. The interface also includes a 'Close' button and a 'Cancel' button at the bottom right.



Purpose statement incorporates motivations

Entity: BED
 Data Asset Type: Principal Data Entity
 Purpose: This is a substructure within the Room substructure of the Facility Location. It contains information about beds within rooms.
 Source: Maintenance Manual for File and Table Data (Software Version 3.0, Release 3.1)
 Attributes: Bed.Description
 Bed.Status
 Bed.Sex.To.Be.Assigned
 Bed.Reserve.Reason
 Associations: >0-+ Room
 Status: Draft

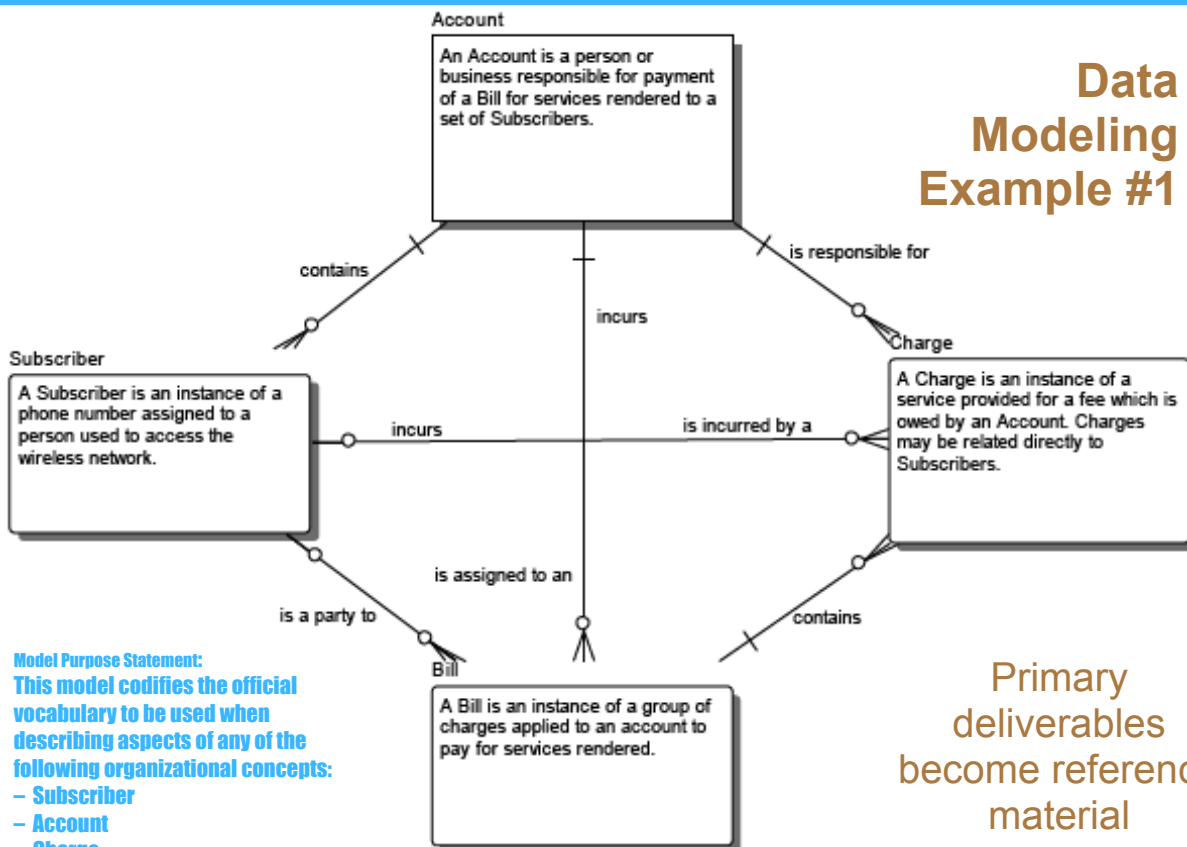
Missing Bed.Transponder.Id?

A purpose statement describing

- Why the organization is maintaining information about this business concept;
- Sources of information about it;
- A partial list of the attributes or characteristics of the entity; and
- Associations with other data items (read as "One room contains zero or many beds.")



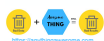
Data Modeling Example #1



Model Purpose Statement:
 This model codifies the official vocabulary to be used when describing aspects of any of the following organizational concepts:

- Subscriber
- Account
- Charge
- Bill

Primary deliverables become reference material

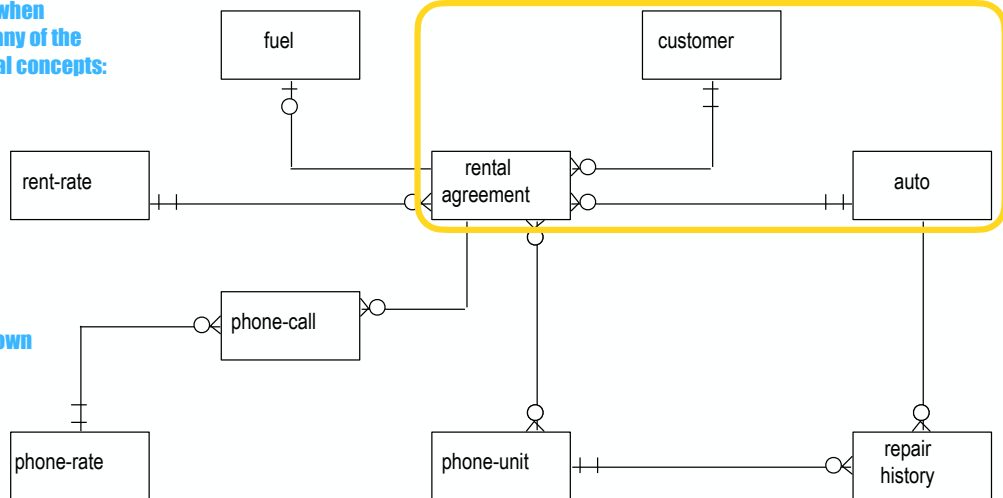


Data Modeling Example #2

Model Purpose Statement:
This model codifies the official vocabulary to be used when describing aspects of any of the following organizational concepts:

- fuel
- customer
- auto
- rental agreement
- rent-rate
- phone-call
- phone-rate
- phone-unit
- repair history

It is documentation shown during the on-boarding process



Interpretations:

1. Car rental company
2. Rental agreement is central
3. No direct connection between customer and contract

4. Contract must have a customer

5. Nothing structural prevents autos from being rented to multiple customers

6. Phone units are tied to rentals

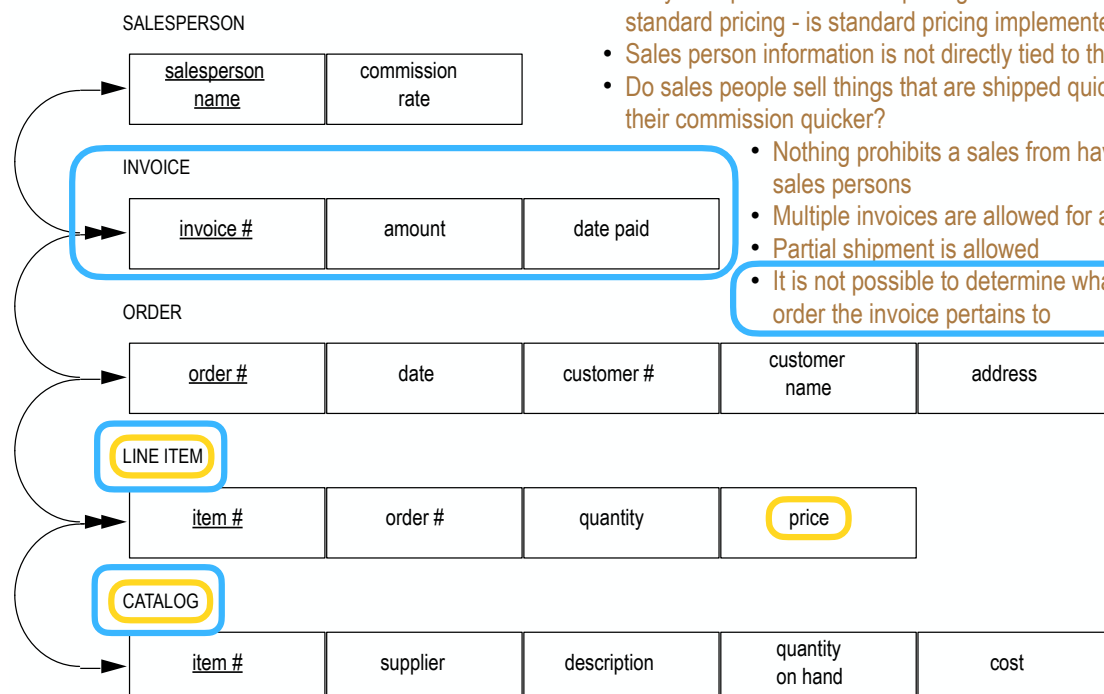


Source: Chikofsky 1990

© Copyright 2024 by Peter Allen Slide # 65

Data Modeling Example #3

- Sales commission-based pricing information
- Difficult to change a customer address
- Price not included in the catalog
- Easy to implement variable pricing - difficult to implement standard pricing - is standard pricing implemented
- Sales person information is not directly tied to the order
- Do sales people sell things that are shipped quickly so they get their commission quicker?



- Nothing prohibits a sales from having multiple sales persons
- Multiple invoices are allowed for a single order
- Partial shipment is allowed
- It is not possible to determine what part of an order the invoice pertains to



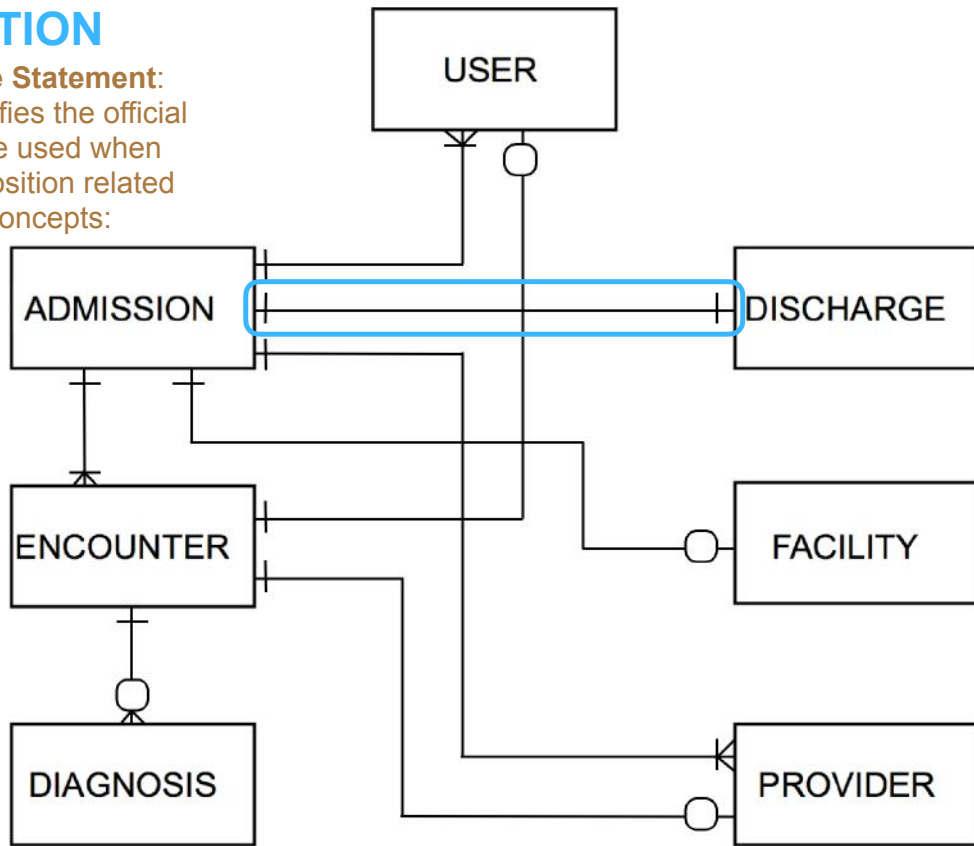
© Copyright 2024 by Peter Allen Slide # 66

Data Model #4: DISPOSITION

Model Purpose Statement:

This model codifies the official vocabulary to be used when describing disposition related organizational concepts:

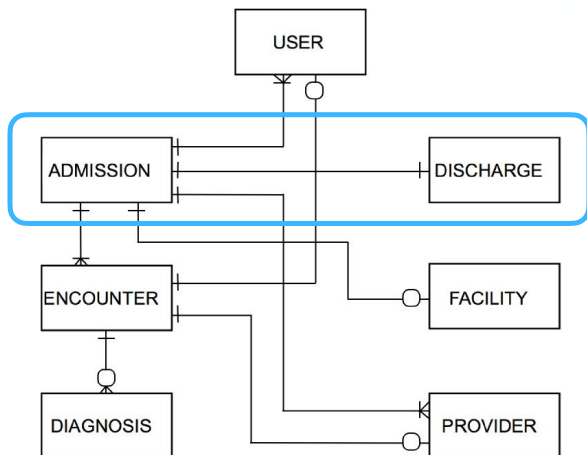
- user
- admission
- discharge
- encounter
- facility
- provider
- diagnosis



Data Model #4: DISPOSITION

- At least one but possibly more system USERS enter the DISPOSITION facts into the system.
- An **ADMISSION is associated with one and only one DISCHARGE.**
- An ADMISSION is associated with zero or more FACILITIES.
- An ADMISSION is associated with zero or more PROVIDERS.
- An ADMISSION is associated with one or more ENCOUNTERS.
- An ENCOUNTER may be recorded by a system USER.
- An ENCOUNTER may be associated with a PROVIDER.
- An ENCOUNTER may be associated with one or more DIAGNOSES.

Death must be a valid disposition code!

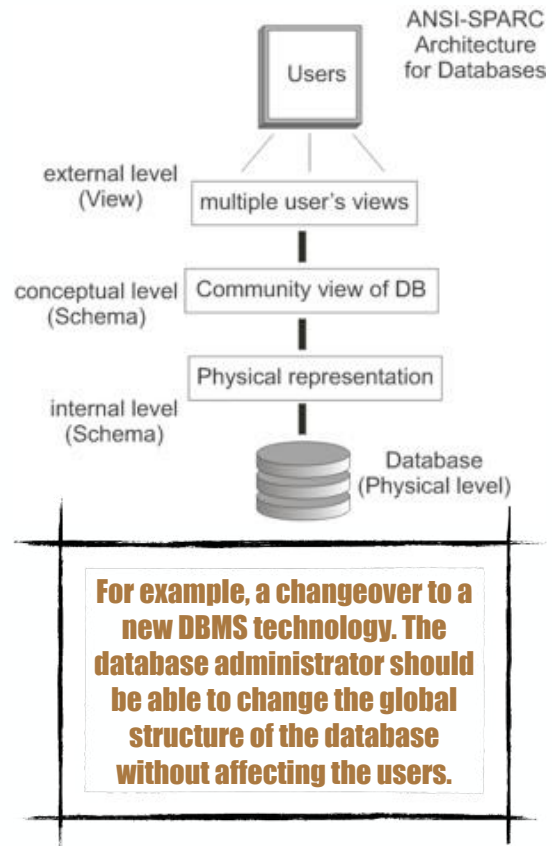


- ADMISSION** Contains information about patient admission history related to one or more inpatient episodes
- DIAGNOSIS** Contains the International Disease Classification (IDC) of code representation and/or description of a patient's health related to an inpatient code
- DISCHARGE** A table of codes describing disposition types available for an inpatient at a FACILITY
- ENCOUNTER** Tracking information related to inpatient episodes
- FACILITY** File containing a list of all facilities in regional health care system
- PROVIDER** Full name of a member of the FACILITY team providing services to the patient
- USER** Any user with access to create, read, update, and delete DISPOSITION data

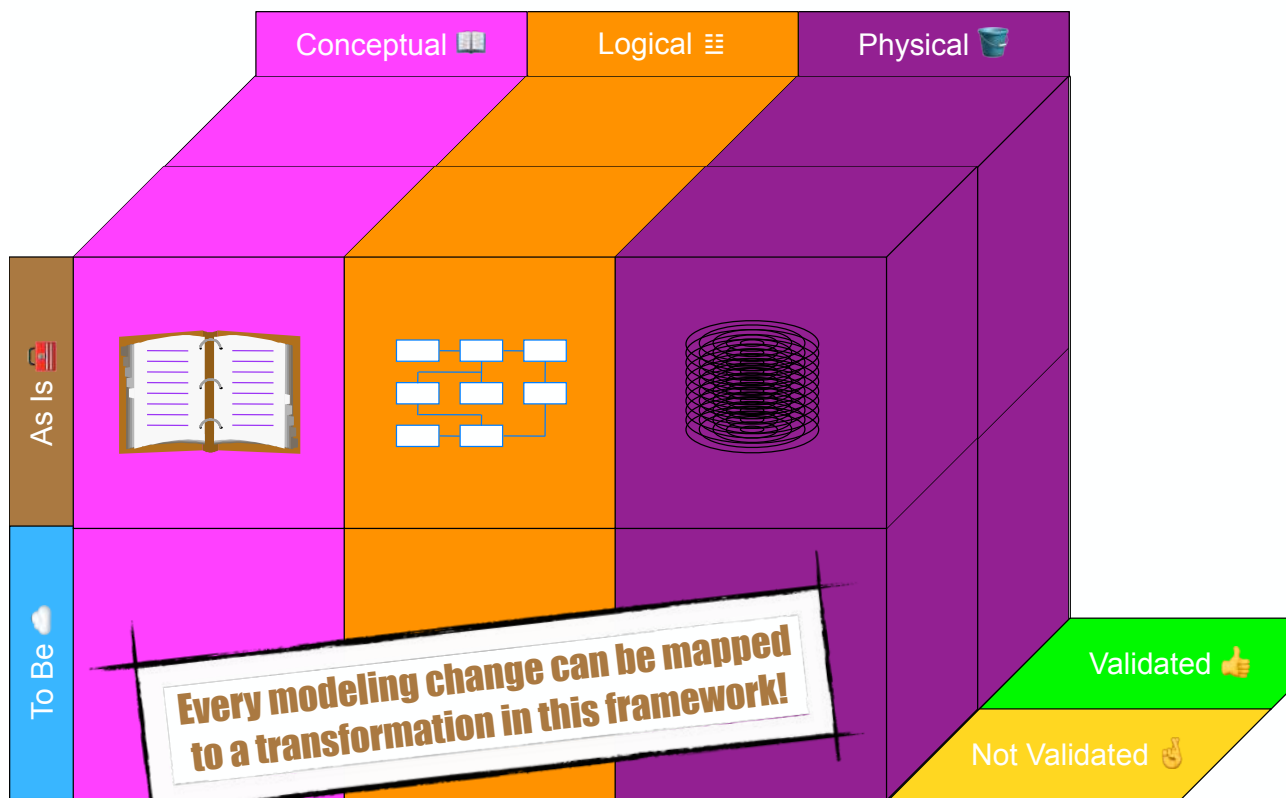


ANSI-SPARC 3-Layer Schema

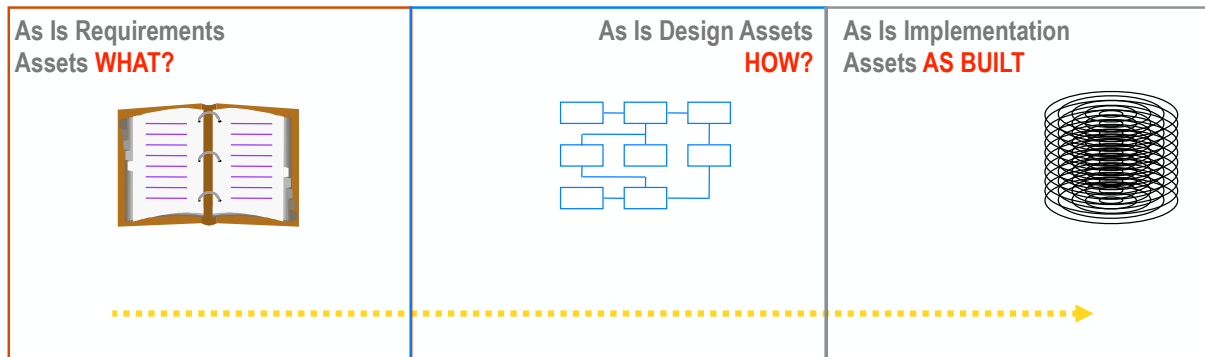
1. **Conceptual** - Allows independent customized user views:
 - Each should be able to access the same data, but have a different customized view of the data.
2. **Logical** - Hides the storage details from users:
 - Users should not have to deal with physical database storage details. They should be allowed to work with the data itself, without concern for how it is physically stored.
3. **Physical** - The database should be able to change the database storage structures without affecting the users' views:
 - Changes to the structure of an organization's data will be required. The internal structure of the database should be unaffected by changes to the storage



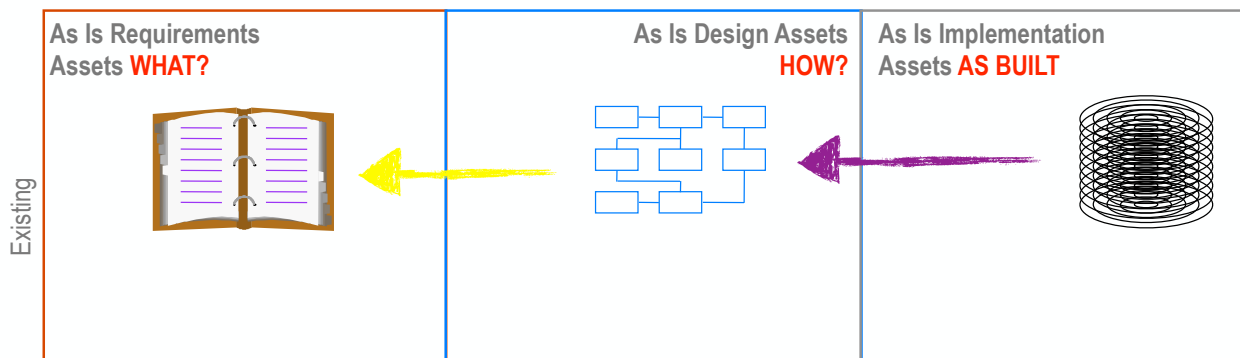
3-Dimensional Model Evolution Framework



Forward Engineering

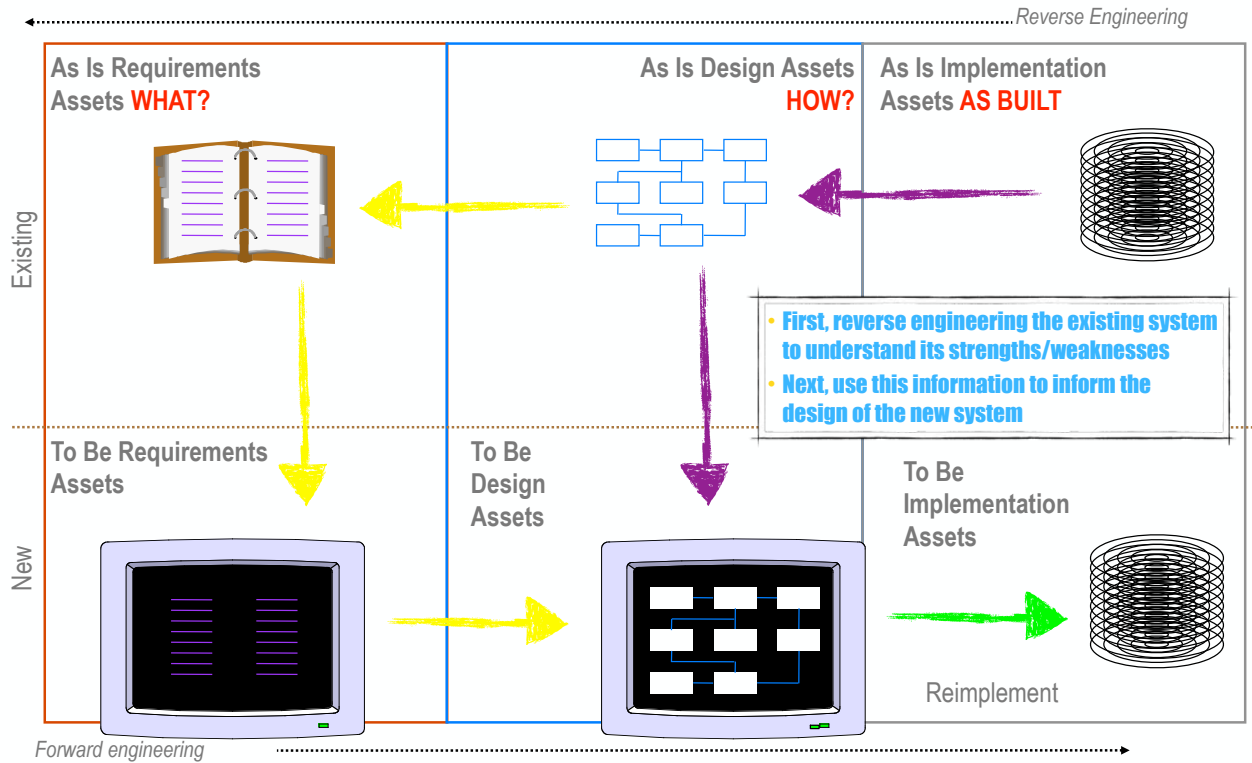


Reverse Engineering

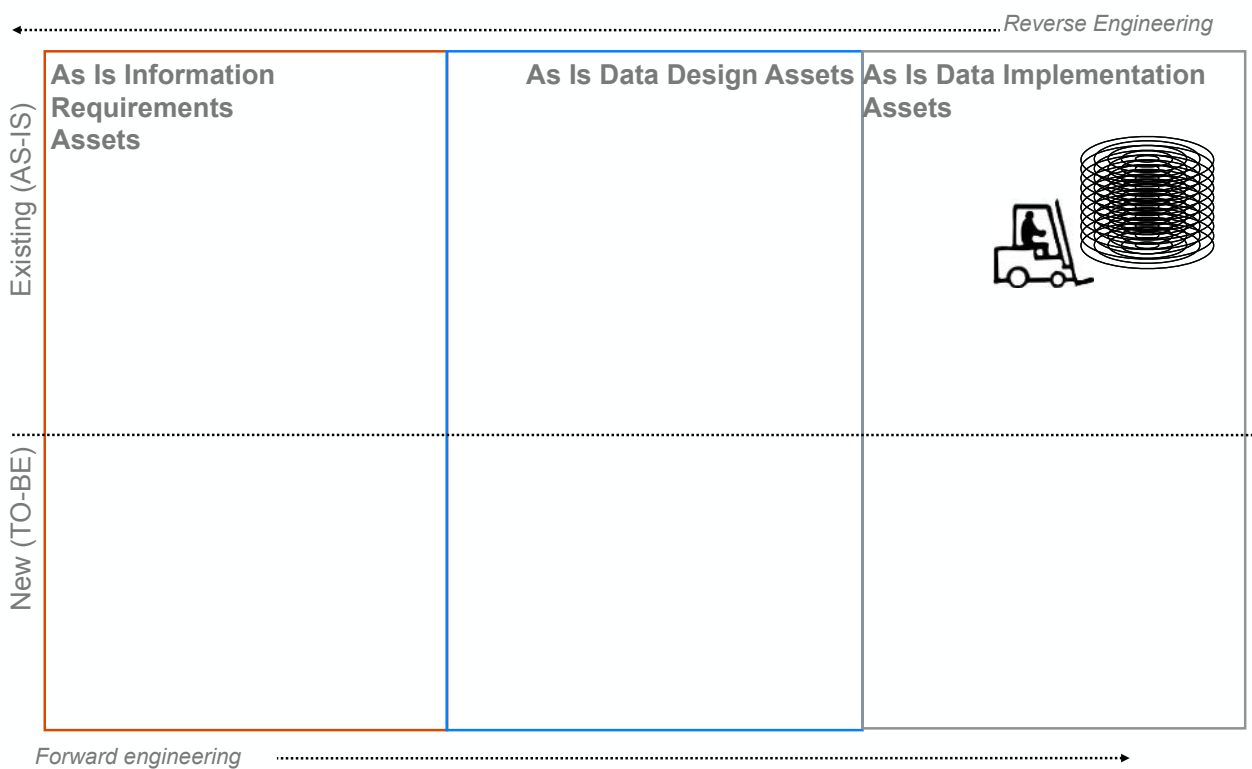


A structured technique aimed at recovering rigorous knowledge of the existing system to leverage enhancement efforts (Chikofsky & Cross 1990)

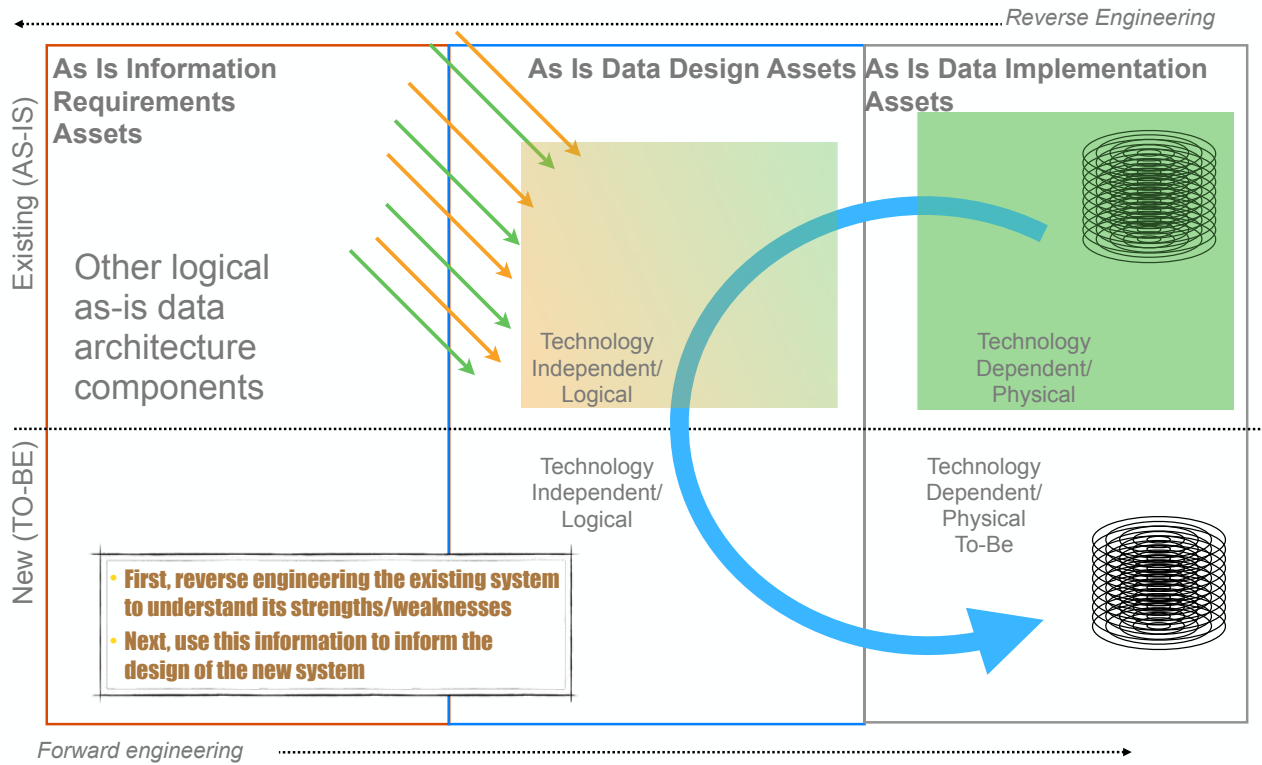
Reengineering



Normalization (how not to do it but how it is always done)



Normalization (better explanation)

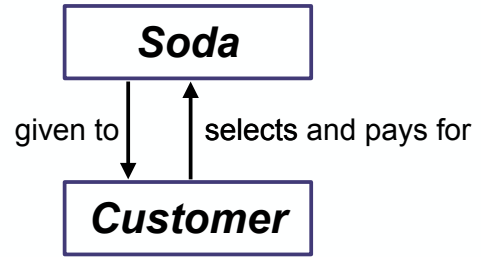


Each modeling cycle has an articulated purpose



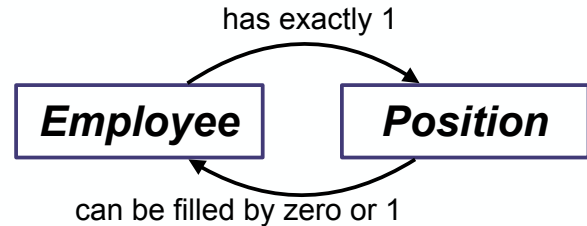
Keep them focused on data model purpose

- The reason we are locked in this room is to:
 - Mission: *Understand formal relationship between soda and customer*
 - Outcome: Walk out the door with a data model this relationship
 - Mission: *Understand the characteristics that differ between our hospital beds*
 - Outcome: We will walk out the door when we identify the top three characteristics required to manage hospital beds.
 - Mission: *Could our systems handle the following business rule tomorrow?*
 - "Is job-sharing permitted?"
 - Outcomes: Confirm that it is possible to staff a position with multiple employees effective tomorrow



Bed
Entity: BED
Purpose: This is a substructure within the room substructure of the facility location. It contains information about beds within rooms.
Attributes: Bed.Description, Bed.Status, Bed.Sex.To.Be.Assigned, Bed.Reserve.Reason
Associations: >0-+ Room
Status: Validated

How does our perspective change: *the primary means of tracking a patient*



Don't Tell Them That You Are Modeling!

Just write some stuff down

Then arrange it

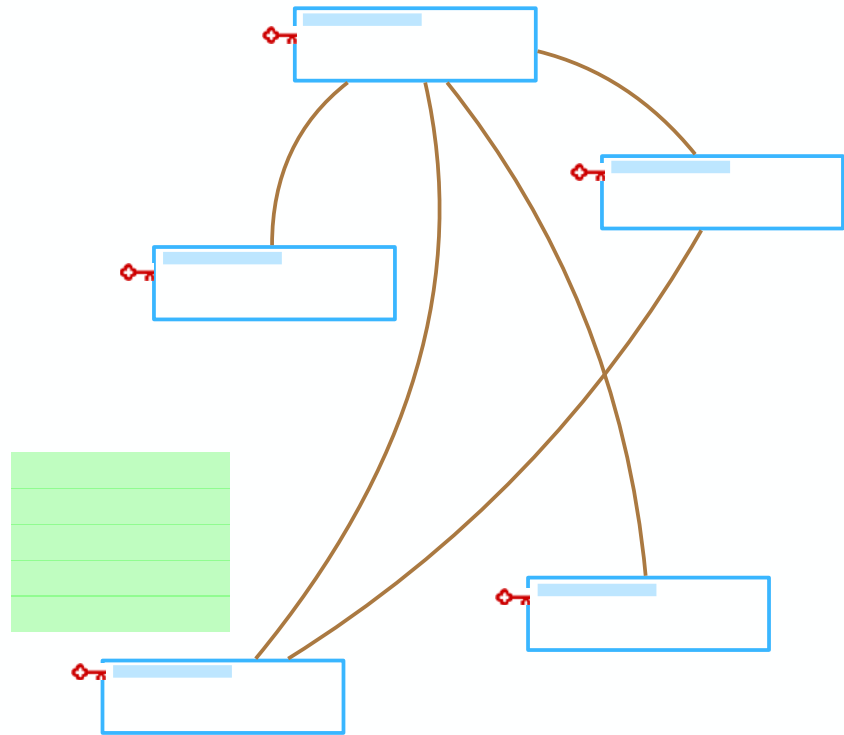


Then make some appropriate connections between your objects



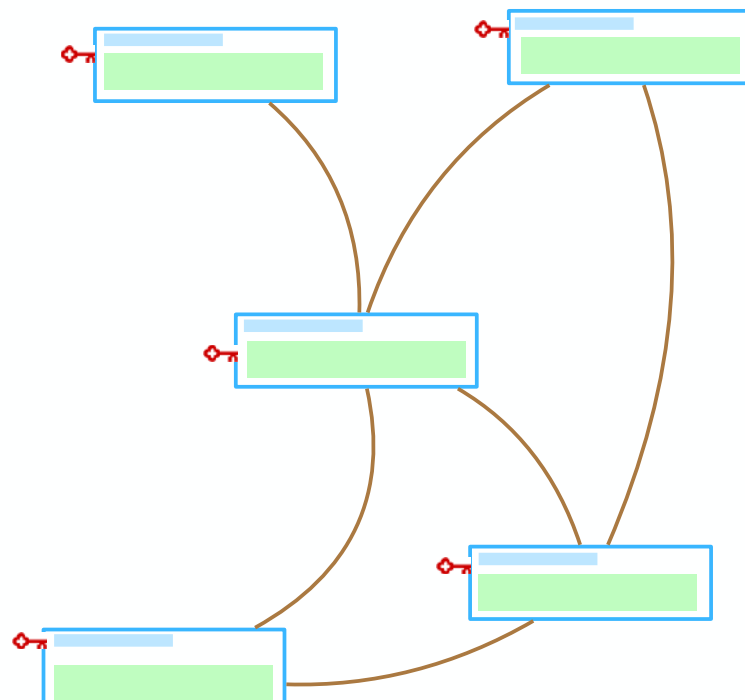
Data Modeling Process

1. Identify entities
2. Identify key for each entity
3. Draw rough draft of entity relationship data model
4. Identify data attributes
5. Map data attributes to entities

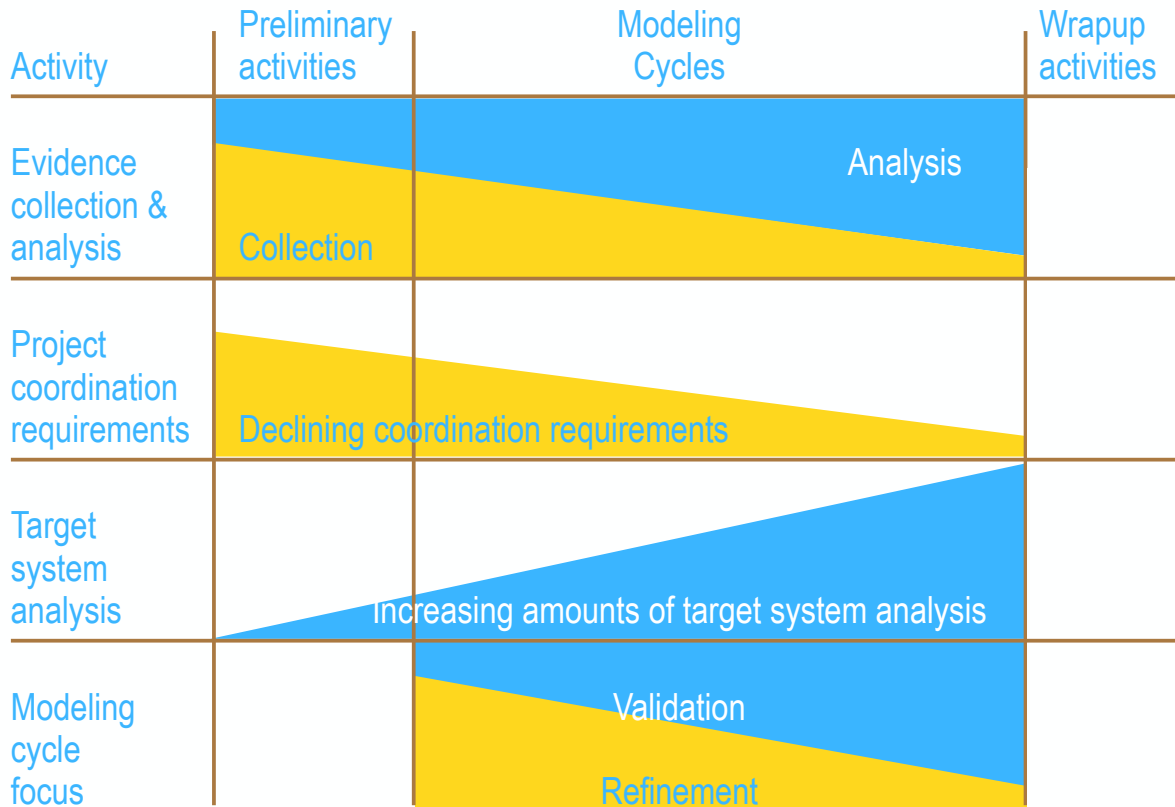


Model evolution is good, at first ...

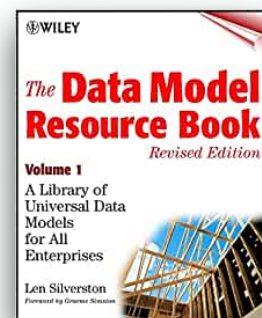
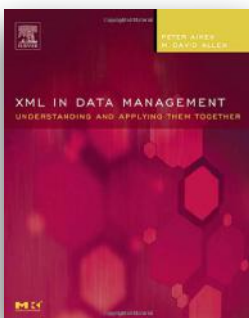
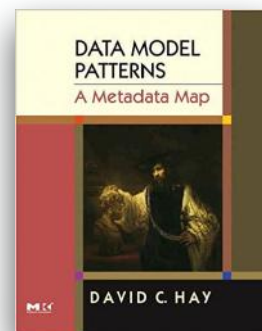
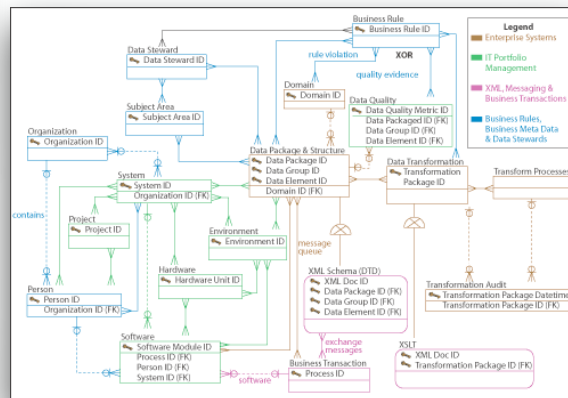
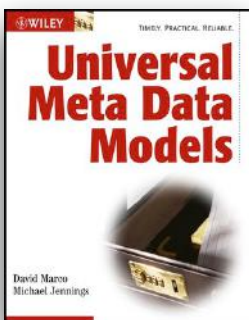
1. Identify entities
2. Identify key for each entity
3. Draw rough draft of entity relationship data model
4. Identify data attributes
5. Map data attributes to entities



Relative use of time allocated to tasks during modeling



Metadata Models

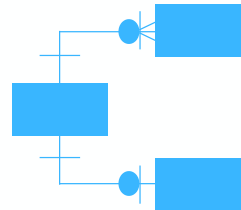


Program Overview

- What is data modeling required for?
 - Increasing understanding—systems to humans
 - Precisely defining data
 - Achieving simplification goals
 - Focused points of agreement
 - Understanding, building & deploying stable business models/strategy
- Why is modeling required for data understanding?
 - Why model anything?
 - Data requires precise definitions/agreements/understandings
 - Suboptimal data practices accumulate data debt
 - Data modeling describes important details about data that must be perfect
 - Should be considered from a primarily iterative development context
 - Understand and document data structures
- How using data models effectively
 - Leverages a standard communication form
 - Reduces corrosive anomalies
 - Keeps focus on motivation using purpose statements
 - Captures and communicates vital business information
 - Forward engineering (Goal=Development/Building)
 - Reverse engineering (Goal=Understanding)
- Take Aways, References, Q&A



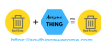
Data Modeling Fundamentals
Achieving common understanding



© Copyright 2014 by Peter Allen Slide # 83

Data Modeling for Business Value

- Goal must be shared IT/business understanding
 - No disagreements = insufficient communication
- Data sharing/exchange is automated and dependent on successful engineering/architecture
 - Requires a sound foundation of data modeling basics (the essence) on which to build technologies
- Modeling characteristics evolve during the analysis
 - Different model instances may be useful to different analytical problems
- Incorporate motivation (purpose statements) in all modeling
 - Modeling is a **problem defining** as well as a **problem solving activity**
- Use of modeling is more important than selection of a specific method
- Models are often living documents
- Models need to be available in an easily searchable manner
- Utility is paramount
 - Adding color and diagramming objects customizes models and allows for a more engaging and enjoyable user review process

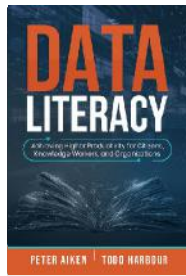


Inspired by: Karen Lopez http://www.information-management.com/newsletters/enterprise_architecture_data_model_ERP_BI-10020246-1.html?pg=2

© Copyright 2014 by Peter Allen Slide # 84

Event Pricing

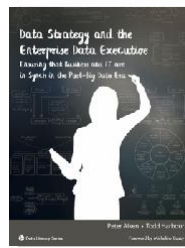
- 20% off directly from the publisher on select titles
- My Book Store @ <http://anythingawesome.com>
- Enter the code "anythingawesome" at the Technics bookstore checkout where it says to "Apply Coupon"



Data Literacy: Achieving Higher Productivity for Citizens, Knowledge Workers, and Organizations

Citizens and organizations need to improve their data literacy to 'do more with data'

[Learn More](#)



Data Strategy and the Enterprise Data Executive

Ensuring that Business and IT are in Synch in the Post-Big Data Era

[Learn More](#)



The CDO Journey

Insights and Advice for Data Leaders

Note: the authors have donated all proceeds from this title towards the International Society for Chief Data Officers.

[Learn More](#)



Monetizing Data Management

17 Case Studies illustrating How Data Leveraging (Big and Small) Can Produce Quantifiable Results That Are of Keen Interest to C-Suite Occupants

[Learn More](#)

[anythingawesome](#)

[Apply coupon](#)



© Copyright 2024 by Peter Aiken Slide 85

Upcoming Events

Time: 19:00 UTC (2:00 PM NYC) | Presented by: Peter Aiken, PhD

Data Stewards: Defining and Assigning 12 March 2024

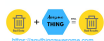
Essential: Reference and Master Data Management 9 April 2024



The Core Concepts of Data Ethics 14 May 2024

Brought to you by:

[Clicking any webinar title will link directly to the registration page]



© Copyright 2024 by Peter Aiken Slide 86

Independent Verification & Validation

Critical Design Review?

Mentoring?

Collaboration?

Executive Data
Literacy Training?

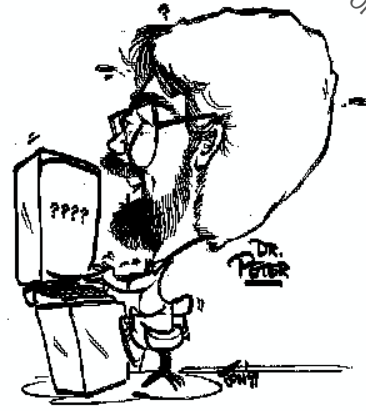


+



STILL

=



Peter.Aiken@AnythingAwesome.com +1.804.382.5957

Reverse Engineering Expertise?

Hiring Assistance?

Thank You!

Use your data more strategically?

Tool/automation evaluation?

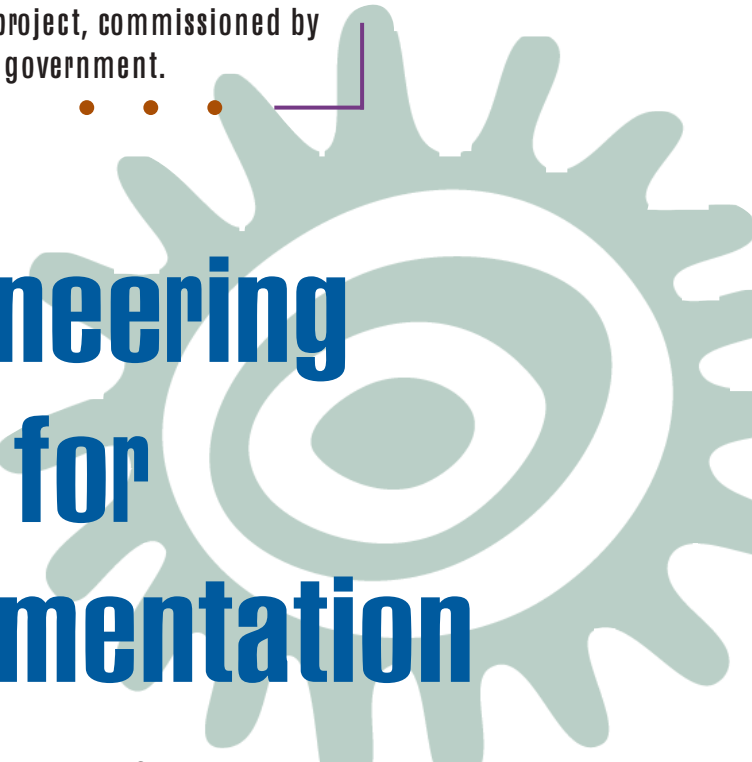


Book a call with Peter to discuss anything - <https://anythingawesome.com/OfficeHours.html>

Case Study

Reverse-engineering a commercial client-server system from PeopleSoft yielded a valuable resource and proved to be cost-effective. The authors describe the motivations for, approach to, and results of this project, commissioned by the Commonwealth of Virginia's government.

Reverse-Engineering New Systems for Smooth Implementation



Peter Aiken and Ojelanki K. Ngwenyama, Virginia Commonwealth University
Lewis Broome, Innovative Business Solutions

The Commonwealth of Virginia's departments of Personnel and Training (DP&T) and Accounts (DOA) undertook a \$12 million effort in 1994 to replace their existing payroll and personnel information systems because the technology was dated, it no longer provided timely information support, and the databases were not integrated. Its systems had become inefficient and cost-prohibitive to operate and maintain, and management was concerned about keeping qualified technical expertise on staff to support them. Successful implementation of a new integrated human resource information system, or IHRIS, required top management to be sensitive to public-sector funding pressure and the political fallout of government program failures.^{1,2} (More information is available at <http://fast.to/peteraiken/>.) As a result, the transition team had to show rapid and credible progress implementing the new system.

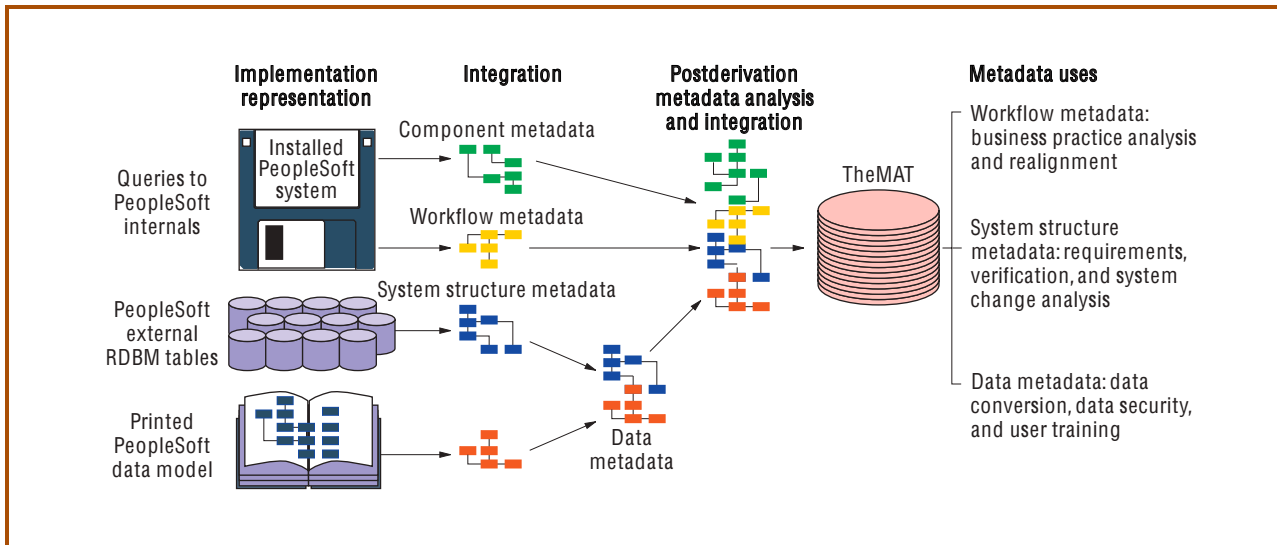


Figure 1. Reverse engineering, sources used to derive metadata, and its uses.

By 1997, management selected the PeopleSoft Human Resources, Benefits, and Pay modules as the basis for the new IHRIS system design. The client software was designed to run on PC desktops accessing a superserver providing relational data management and distributed workgroup servers hosting associated utilities. The PeopleSoft system was developed specifically for distributed client-server implementation, but the company expected the modules to be tailored to fit the needs of each organization.

To leverage the investment in its existing system, the DP&T/DOA decided to reverse-engineer the PeopleSoft modules. The initial motivation was to develop a detailed understanding of the modules in order to convert the two legacy systems' data. The goal of reverse engineering is to analyze a system to uncover facts about its design and functionality, often called metadata, to facilitate development and implementation activities. Metadata, or "data about data," is defined by the ISO description 11179 as "the information and documentation which makes data sets understandable and sharable for users."³ It also describes facts about the system components.⁴

Organizations concerned with effective and efficient operations can benefit tremendously from easily accessible system metadata. From a practitioner perspective, new systems are likely to be more easily reverse-engineered as system developers adopt architecture-based approaches to development. This architecture provided the IHRIS transition team with easily accessible and obtainable system metadata with a modest investment.

GETTING STARTED

An innovative industry-academic partnership

between the DP&T/DOA and the Virginia Commonwealth University's Information Systems Research Institute was formed to establish part of the IHRIS transition team. In the Spring of 1997, a reverse engineering team was organized to reverse-engineer the PeopleSoft modules and obtain the metadata. The RET consisted of four graduate students, each of whom had studied advanced analysis and design and database management. By August 1997, the team had provided more than 1,200 professional and 6,000 student project hours in support of IHRIS implementation. The RET logged 660 hours of total project time between April and September 1997 for an approximate total project cost of \$21,000 (600 student hours at \$20 per hour and 60 project leader hours at \$150 per hour).

Motivated by the need to better understand the system, the RET applied reverse engineering as a systematic approach to examine, document, model, and analyze the system. The team followed an iterative, evolutionary approach to derive the metadata; this was necessary because the process of analyzing and understanding the system was itself iterative. The RET would often begin with an understanding of part of the system, only to have that understanding evolve as new information emerged. The reverse-engineering analyses consisted of three steps applied repeatedly.

- ◆ Identify and describe a metadata requirement.
- ◆ Identify, derive, and integrate the metadata.
- ◆ Use the metadata to support an implementation task.

Figure 1 indicates graphically how these analyses were carried out and how the metadata was derived, analyzed, and integrated. Three metadata sources were the system itself, the system documentation, and other nonsystem sources.

Table 1
Comparison of System Characteristics

System	Platform	OS	Age in 1998	Structure	Approximate Number of Data			
					Physical Records	Logical Records	Entities	Attributes
Payroll (LS1)	Amdahl	MVS	15	VSAM/virtual database tables	780,000	60,000	4/350	683
Personnel (LS2)	Unisys	OS	21	DMS (network database)	4,950,000	250,000	57	1,478
PS-L	PC	Win 95	New	Client-server RDBMS	600,000	250,000	1,600	15,000
PS-P	PC	Win 95	New	Client-server RDBMS	600,000	250,000	2,706	7,073

SYSTEM CHARACTERISTICS

The two systems to be replaced were used by DP&T/DOA to merge payroll and personnel records. The payroll system (LS1) was a partially implemented commercial system with in-house extensions. It processed the payroll for more than 130,000 state employees and was designed around four large VSAM files. At runtime, data was redefined by the system to appear as more than 350 unique, virtual database tables. The personnel system (LS2) was developed completely in-house using a platform-specific database, associated Mapper, and other software programs. The LS2 database maintained personnel benefits records for roughly 100,000 full-time employees and more than 50,000 part-time employees (130,000 of which had corresponding LS1 records), and about 100,000 retirees.

The data architecture of the PeopleSoft modules employs a relational data model and SQL to ensure compatibility with a number of commercial relational database management systems. The user interface employs rectangular data arrays and manipulation tools. Once users are taught how to use these tools, they can search for their data, within security constraints. The system also provides a user-friendly report generator that supports a wide range of data access, analysis, and presentation facilities.

Table 1 presents a comparison of some basic logical and physical characteristics of LS1, LS2, and the logical and physical models of the PeopleSoft application (PS-L and PS-P respectively).

EXTRACTING, MANAGING, AND INTEGRATING THE METADATA

Most of the reverse engineering was accom-

plished using the PeopleSoft system and a toolkit based on the Microsoft Office Suite. To manage the metadata in the most flexible and adaptable electronic format, the RET developed and implemented a metadata repository database called The Metadata Access Tool, or TheMAT, using MS Access and automated and manual procedures. Various utilities were combined using the integrated project management, word processing, spreadsheet, database, and presentation software capabilities of MS Office. TheMAT maintains the metadata for each system and permits data item mapping between systems.

Initially, the RET developed a hierarchical decomposition model of the PeopleSoft system based on its component structure, workflow, and data views. It then used this information to formulate specific queries about system objects, getting PeopleSoft to report metadata about its own structures. Until the RET obtained the desired results, it repeated the cycle of decomposing specific system objects and deriving enough information to structure specific queries to report the metadata. The team then statistically analyzed the metadata to determine its validity and relationships to other metadata. This analysis cycle is described in the boxed text entitled "Reverse Engineering" on p. 40.

Prior to V6, the PeopleSoft modules were delivered with limited CASE tool support and static technical documentation in PDF format. These were inadequate to support the IHRIS implementation. Additional CASE tool support, costing \$ 1,200 per seat, was purchased from another vendor, but this provided only marginally better functionality. To work with these models the RET had to print them, then manually cut and paste more than 200 pages together. More importantly, the paper and PDF formats prohibited browsing or computer-based analysis of the model content or structure. As the project

planning proceeded, requirements for more robust metadata emerged.

METADATA REQUIREMENTS

The RET's first priority was to analyze the transition team needs to understand the new system. The analysis focused on discovering the system core, as did similar investigations.^{5,6} Requirements analysis revealed the need to understand and describe the system structure, workflow, and data metadata. Structure metadata describes user system interaction according to interaction roles. Workflow metadata describes the processes supported by the system. Data metadata describes the data structure and relationships.

The IHRIS transition team wanted electronic metadata access when working individually and in groups. They wanted the ability to manipulate and analyze the metadata and to derive useful information from it when performing implementation tasks. They wanted to understand and describe the system in terms of structure, workflow, and data. These requirements and the metadata volume required a central repository and interactive graphical analysis tools.

Figure 2 illustrates the PeopleSoft enterprise software metadata that describe the Human Resources, Benefits, and Pay modules. This model describes the relationships represented by various types of metadata, the entity definitions, and the number of facts derived by the reverse engineering. The number of facts also indicates relative system component complexity. For example, the *records* and *fields* entities represent 2,705 Oracle tables, 7,973 columns, and 5,873 instances of *fields* occurring on *records* in the physical database. In this figure, the metadata representing *panels*, *menuitems*, *records*, and *fields* connect the three types of metadata.

System structure metadata requirements

One transition team requirement was to understand the system functionality. Understanding structure metadata was key to determining the collection of *panels* supporting specific IHRIS business activities. As implemented by DPT, the PeopleSoft system structure consists of a hierarchical arrangement of menu structures leading to more than 1,400 *panels*. The *panels* are the interfaces where users create, read, update, or delete data. To reach a given *panel*, users must sometimes navigate through a

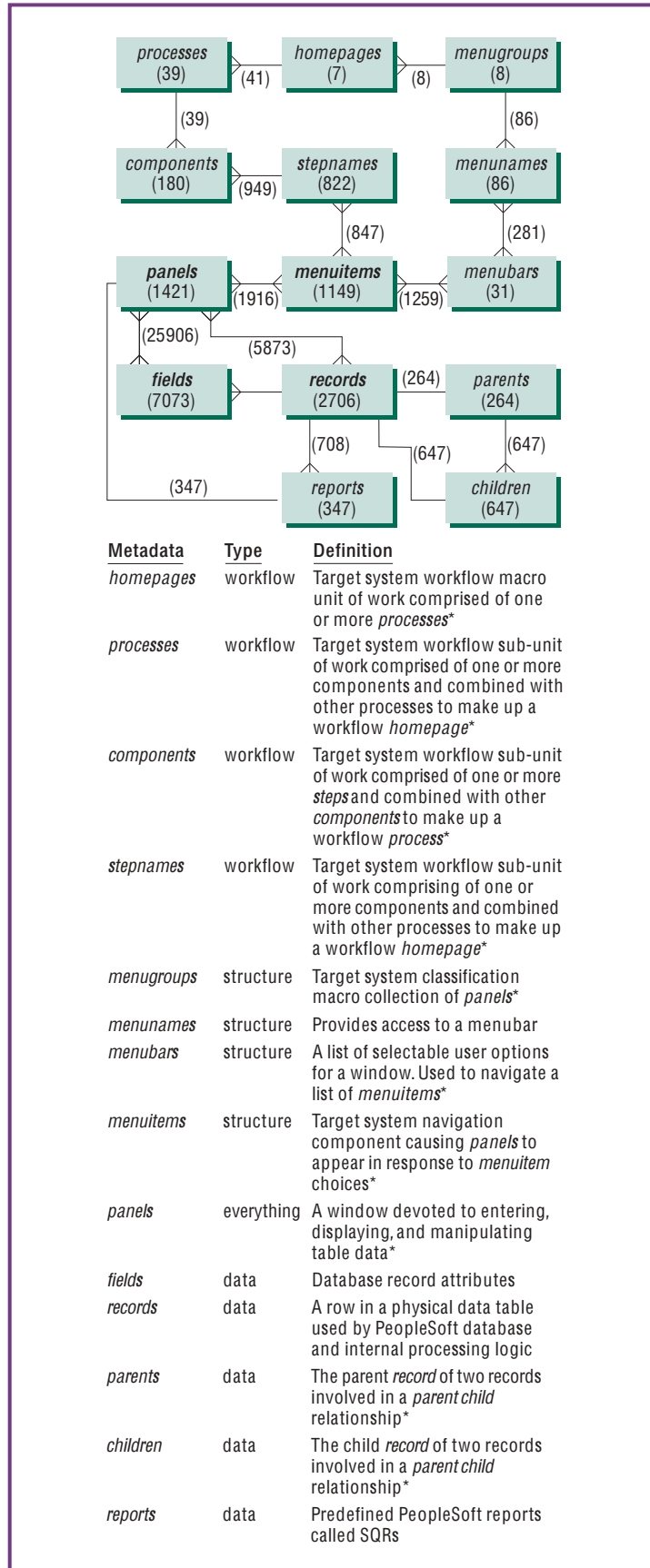


Figure 2. Logical metadata requirements, definitions, and volume. (Note: those definitions followed by (*) are taken directly from PeopleSoft documentation.)

REVERSE ENGINEERING

The Reverse Engineering Team developed a hierarchical decomposition model of the PeopleSoft system based on its component structure, workflow, and data views. This information was used to formulate specific queries about system objects, getting it to report metadata about its own structure. The metadata was then statistically analyzed to determine its validity and relationships to other metadata. The iterative reverse-engineering analysis cycle^{1,2} is summarized in Table A in six steps.

The last step usually resulted in more refined requests for additional metadata, and the cycle began again. This general model of extracting specific information from the system via SQL queries, restructuring it, and integrating it with existing TheMAT contents was repeated with many analysis variations to develop the metadata. The variations generally consisted of changing

the source of the analysis inputs to include different system objects, the system documentation, or the metadata itself. These analyses provided the RET with the facility to report on associations among any set of systems objects using SQL queries. In this way, RET members were able to define and document the PeopleSoft system metadata and relate these to the LS1 and LS2 metadata.

REFERENCES

1. P. Aiken, *Data Reverse Engineering*, McGraw Hill, New York, 1996.
2. P. Aiken, "The Reverse Engineering of Data," *IBM Systems J.*, Vol. 37, No. 2, 1998, pp. 246-269.

Table A

Reverse-Engineering Analysis Steps	Illustration
Determine what type of metadata to derive for specific system implementation requirements.	Implementation needs indicate a requirement to understand the workflow metadata by obtaining a list of all combinations of <i>stepname</i> instances along with each associated <i>component</i> , <i>business process</i> , and <i>homepage</i> .
Formulate a query designed to extract specific metadata.	Using SQL, extract from the system all unique combinations of <i>homepages</i> , <i>processes</i> , <i>components</i> , and <i>stepnames</i> resulting in a 13,044-line report.
Export the extracted metadata to a spreadsheet for subsequent complexity analysis and validation.	Saving the query results into .xls format permits further statistical analysis of the metadata to determine information about metadata relationships, complexity, and occurrence frequency in order to confirm the extracted metadata correctness.
Import the validated metadata into TheMAT.	Once validated, the process metadata is moved into the MS Access database as a new stand-alone table.
Integrate the new metadata with the existing metadata.	The new table is formally associated with the existing metadata, in this case linking processes to <i>menugroups</i> via <i>homepages</i> and <i>stepnames</i> to <i>menubars</i> and <i>panels</i> via <i>menuitems</i> (as shown in Figure 2).
Provide the resulting, richer metadata to the requesting user, verifying the metadata and its utility.	Enhance TheMAT reporting capabilities, publish the next version, and work directly with the user group requesting the metadata to ensure that the metadata is accurate and meets their needs.

four-tier hierarchical menu structure comprised of (from highest to lowest) *homepages*, *menugroups*, *menunames*, *menubars*, and *menuitems*. The resulting metadata supported the transition analysis by providing responses to the following questions:

- ◆ Given a *menuname*, what are the possible *menubar* choices?
- ◆ From what *menuitems* can this *panel* be accessed?

◆ Given a *menubar*, what are its associated *menuitems*?

The associations between higher and lower navigation components are usually one higher-level item related to many lower-level items. However, most implementations required support for many relationships. The RET identified and documented all the relationships among these components.

Workflow metadata requirements

A second implementation requirement focused on aligning the organizational business practices with the new modules' functionality. The workflow metadata provided the transition team with descriptions of the PeopleSoft business processes. The workflow metadata was used to compare the PeopleSoft process structure to the DP&T/DOA's. The metadata supported mappings linking the DP&T/DOA business events to PeopleSoft functionality. This provided a framework supporting the transition team's analysis and system-tailoring activities. This information was important to the transition team as they assessed how well the existing work procedures and practices were aligned with those supported by PeopleSoft. The team performed a gap analysis to determine how much an existing process had to be tailored to fit the new system; it also identified and analyzed gaps and developed solutions, modifying either system functionality or business practice or both. The metadata requirements focused on understanding the workflow structure implemented by IHRIS:

- ◆ Given a *business process name*, what are its *components*?
- ◆ Given a *component name*, what are the steps that form it?
- ◆ Given a *panel*, how many *process components* access it?

The metadata was structured and repeated a generally hierarchical arrangement among workflow, consisting of *homepages*, *processes*, *components*, *stepnames*, and *menuitems*. The *menuitems* link the workflow with the system structure metadata.

Data metadata requirements

Third, the IHRIS transition team needed to derive design information about the physical data architectures of the new system for comparison and mapping to the legacy system data. This metadata was needed to convert the LS1 and LS2 data structures to the PeopleSoft data structures. The LS1 and LS2 data structures were reverse-engineered, and the metadata derived from this process was analyzed and transformed into a standard format for mapping to the PeopleSoft database. It uncovered the need to maintain metadata on more than 2,700 *records* with a subset of more than 7,000 *fields*. This requirement alone represents more than 35,000 facts. Another aspect of the data structure was the 264 *records* that function as *parents* to more than 647 *children*. Because the data metadata also detailed which

panels contained which *fields*, the team could use this data structure to integrate the data, workflow, and structure metadata.

USING THE METADATA IN THE SYSTEM IMPLEMENTATION

The metadata derived by reverse engineering was widely used during the new system implementation. Several examples show how metadata provided valuable system and background information.

The system metadata in TheMAT was used to document discrepancies between capabilities and requirements in the PeopleSoft system. To do this analysis, the transition team demonstrated to users a series of panels that were organized from the system structure metadata. The users associated each system requirement by linking it to corresponding system components. Discrepancies were noted for subsequent investigation and resolution.

The transition team analyzed the discrepancies and evaluated proposed system changes, modifications, and enhancements. Various metadata types were used to assess the magnitude of proposed changes. For example, the IHRIS transition team was interested in the number of panels requiring modification if a given field length was doubled. This information was used to analyze the costs of changing the system versus changing the organizational practices.

Business practice analysis was conducted to identify gaps between the DP&T/DOA business requirements and the PeopleSoft system. The RET mapped the appropriate process components to specific existing user activities and workgroup practices. The mapping helped users focus their attention on relevant portions of the system. For example, the payroll clerks used the metadata to determine which panels belonged to them.

Business practice realignment addressed gaps between system functionality and existing work practices. Once users understood the system's features and could navigate through it, they compared the system's inputs and outputs with their own needs. If gaps existed, they used the metadata to assess the magnitude of proposed changes. This information was then used to forecast the cost of further customizing the system. In a number of instances, these forecasts provided justification for changing the business practice instead of the system.

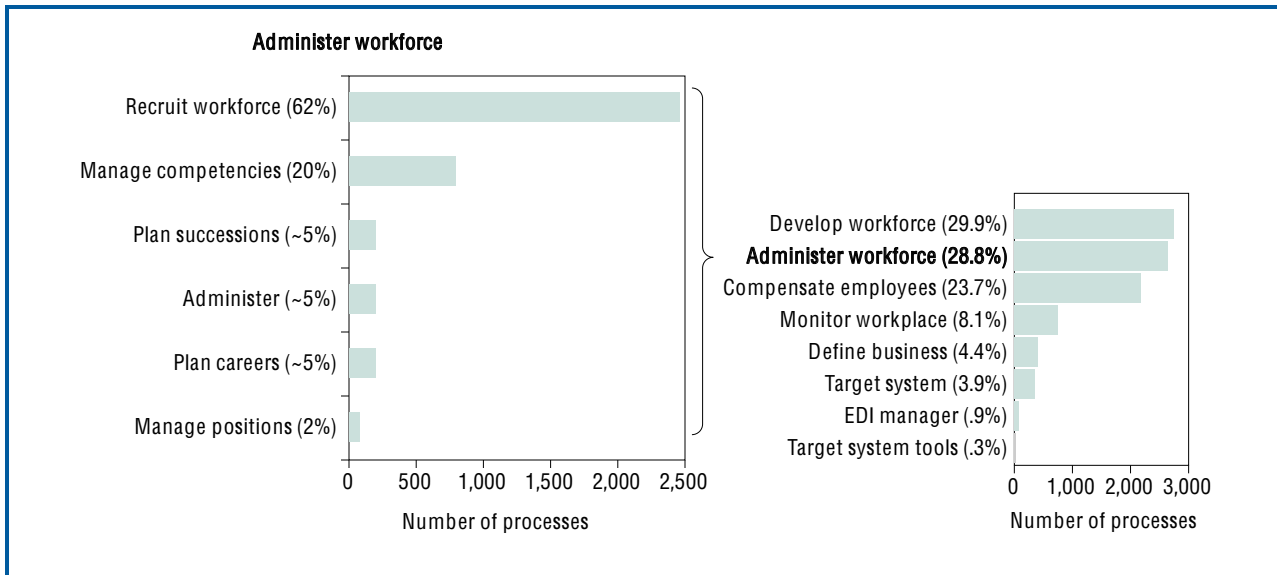


Figure 3. A statistically derived introduction to the PeopleSoft Administer Workforce module business process showing the complexity of the six components and how they fit into the homepage structure.

User training specialists also used the mappings between business practices and system functions to determine which combinations of *panels*, *menuitems*, and *menubars* were relevant to each user group. TheMAT was used to display panels in the sequence expected by the system users. By reviewing panels, users were able to swiftly become familiar with their areas. Additional capabilities for screen session recording and playback were integrated into the toolkit to permit development of system/user interaction "movies" and development of system test scripts.

Several additional metadata were incorporated into TheMAT that are not illustrated in Figure 2, including metadata describing LS1 and LS2, associations with system batch reporting programs, and user and user type metadata. The first supported the data conversion effort and assisted the transition team in developing physical data models. The data conversion subtask was the initial motivation for the metadata development work. More than 300,000 logical and millions of physical records had to be converted from LS1 and LS2 into formats usable by IHRIS. Each decision to convert a data item was recorded, permitting the tracking of the number of data items that had been mapped and converted. The metadata was queried to determine whether a specific data item had been associated with one or more IHRIS data fields, when the conversion had been accomplished, and often what code was used to accomplish the actual conversion.

The metadata helped the team to systematically organize the analysis of the PeopleSoft physical database design. A CASE tool, Visible Advantage, was integrated to extract the database design information directly from the physical database and integrate it into TheMAT, simplifying project documentation. Metadata was used to support the decomposition of the physical database into logical user views. We collected additional metadata to document how the system implements user requirements. This enabled us to track the status of individual requirements. Similarly, metadata was used as the basis for planning security access levels and privileges. Once we integrated the user metadata into TheMAT, we were able to use the system *menuitem/panel* hierarchy as the basis for defining database security views, each granting or denying *menuitem* or *panel* access according to various user profiles.

Statistical analysis was also useful for guiding metadata-based data integration from the two legacy systems. For example, the data metadata was used by the IHRIS transition team to map the legacy system data into PeopleSoft data structures. Statistical metadata summaries were also used by the RET to describe the system to users. For example, Figure 3 illustrates use of workflow metadata showing the number of processes in each homepage and the Administer Workforce PeopleSoft modules. It indicates the number of *components* associated with each homepage and that the Recruit Workforce was

the most complex component. These charts were used to show the Administer Workforce users why the recruiters were receiving separate training based on the relative complexity of the components comprising the Administer Workforce process.

The IHRIS project showed the effectiveness of managing a lot of information by maintaining a relatively small amount of metadata. While each of the three metadata types had independent value to the transition team, the sheer volume of system facts discouraged rapid comprehension. The integrated metadata made the system information accessible, encouraging implementation personnel to incorporate it as a regular part of their situation analysis and resolution activities. Storing the metadata in its most flexible and adaptable format permits different types of metadata to be integrated, bringing economies of scale to metadata management.

Based on this case study, we have found that the collection and management of metadata is not as expensive as one would expect. Some metadata can be maintained using CASE or metaCASE tools—especially if CASE tools were used to develop the system. Integrated metadata repositories permit systematic and flexible manipulation, management, and control of physical and logical data via SQL and graphical browsers. System metadata that is easy to create and maintain can be a valuable organizational asset. Reasonable investments in targeted reverse engineering of new systems can provide metadata that can be immediately reused as the systems are reengineered, greatly facilitating implementation. ❖

ACKNOWLEDGMENTS

This investigation is sponsored by the Virginia Department of Personnel & Training. Bill Girling, manager of systems, saw the need and developed the initiative. We thank Information Systems Research Institute research associates Leslie Borman, Sasipa Chankaoropkhun, Pawan Pavichitr, and Kim Boos for their professional contributions to IHRIS, ISRI research associates Lynda Hogdson and Sirirat Taweewattanaprecha for their support in preparing this article, and Pat Krebs at PeopleSoft for a technical review of the materials.

REFERENCES

1. S. Hsu, "Welfare Overhaul by Virginia Faces Computer Glitch: Massive Automation Program Behind Schedule; Costs Mount," *Washington Post*, 24 June 1996, p. A-1.
2. M. Hardy, "VA. Drops Pact on Medicaid: EDS' Failure to Finish System Blamed," *Richmond Times Dispatch*, 25 April 1997, p. A-1.
3. ISO 11179:195-1996 Information Technology - Specification and Standardization of Data Elements.

4. C. Hsu et al., "Metadatabase Modeling for Enterprise Information Integration," *J. Systems Integration*, Vol. 2, No. 1, Feb. 1992, pp. 5-37.
5. C. Hsu et al., "Core Information Model: A Practical Solution to Costly Integration Problems," *Computers and Industrial Engineering*, Vol. 1, 1994, pp. 75-83.
6. S. Weibel et al., "An Element Set to Support Resource Discovery: The State of the Dublin Core," *Intl. J. Digital Libraries*, Vol. 1, Issue 2, Jan. 1997, pp. 176-186.

About the Authors



Peter Aiken is a research director with Virginia Commonwealth University's Information Systems Research Institute. He was a computer scientist with the Defense Information Systems Agency from 1992 to 1997. His main interests are systems engineering, systems requirements, data reverse engineering, and

hypermedia-based software requirements engineering tools and techniques.

Aiken received a PhD from George Mason University.



Ojelanki K. Ngwenyama is an associate professor of information systems at Virginia Commonwealth University and a research professor at Aalborg University, Denmark. His research focuses on understanding how people in everyday social activity appropriate and innovate upon information technology to solve relevant problems.

Ngwenyama received an MBA from Syracuse University, an MS in computer and information sciences from Roosevelt University, and a PhD in computer sciences and information systems from the State University of New York.



Lewis Broome is a business analyst for Innovative Business Solutions inc. (IBSI) in Richmond, Virginia. He specializes in the application of information engineering to private businesses and public-sector entities.

Broome received an MS in business information systems from Virginia

Commonwealth University.

Address questions about this article to Aiken at Virginia Commonwealth University, Department of Information Systems, 1015 Floyd Avenue - Room 4170, Richmond, VA 23284-4000; paiken@computer.org.

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 be 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.)

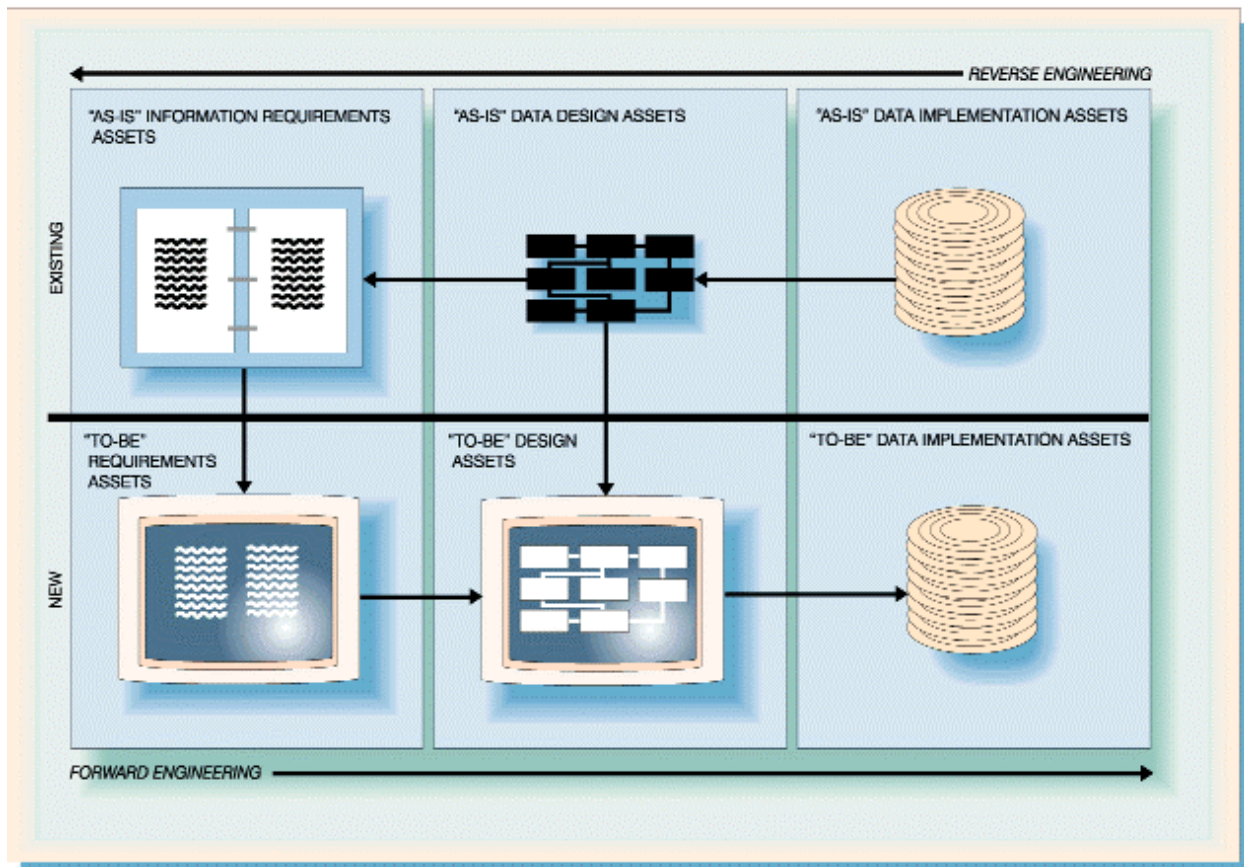
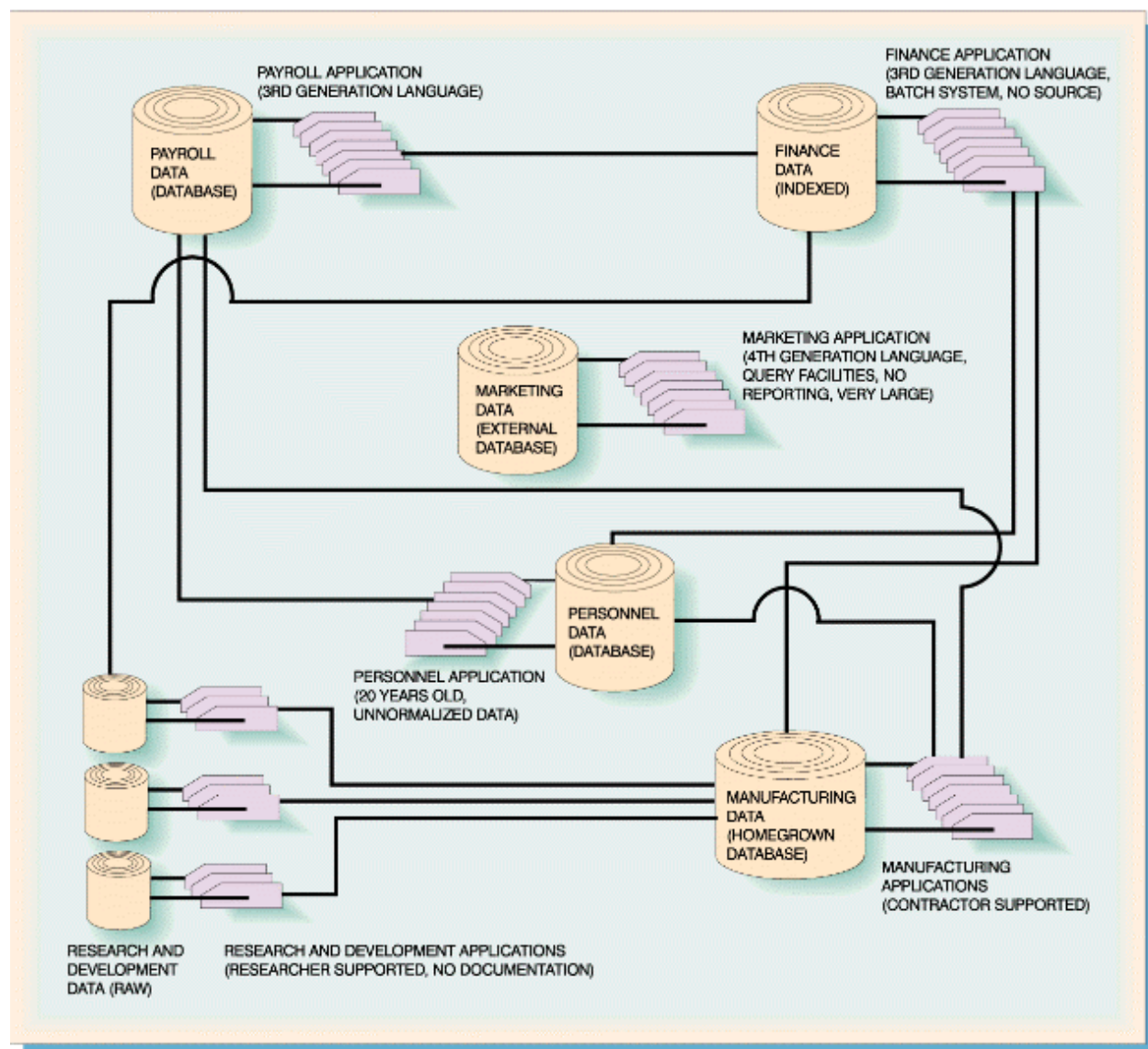


Figure 1 illustrates how SR is based on coordinated reverse and forward engineering activities, where forward engineering benefits from information gained by reverse engineering. A general SR goal is often stated as - delivering output meeting user requirements, using systems that currently do not. Organizations 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's³ 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.

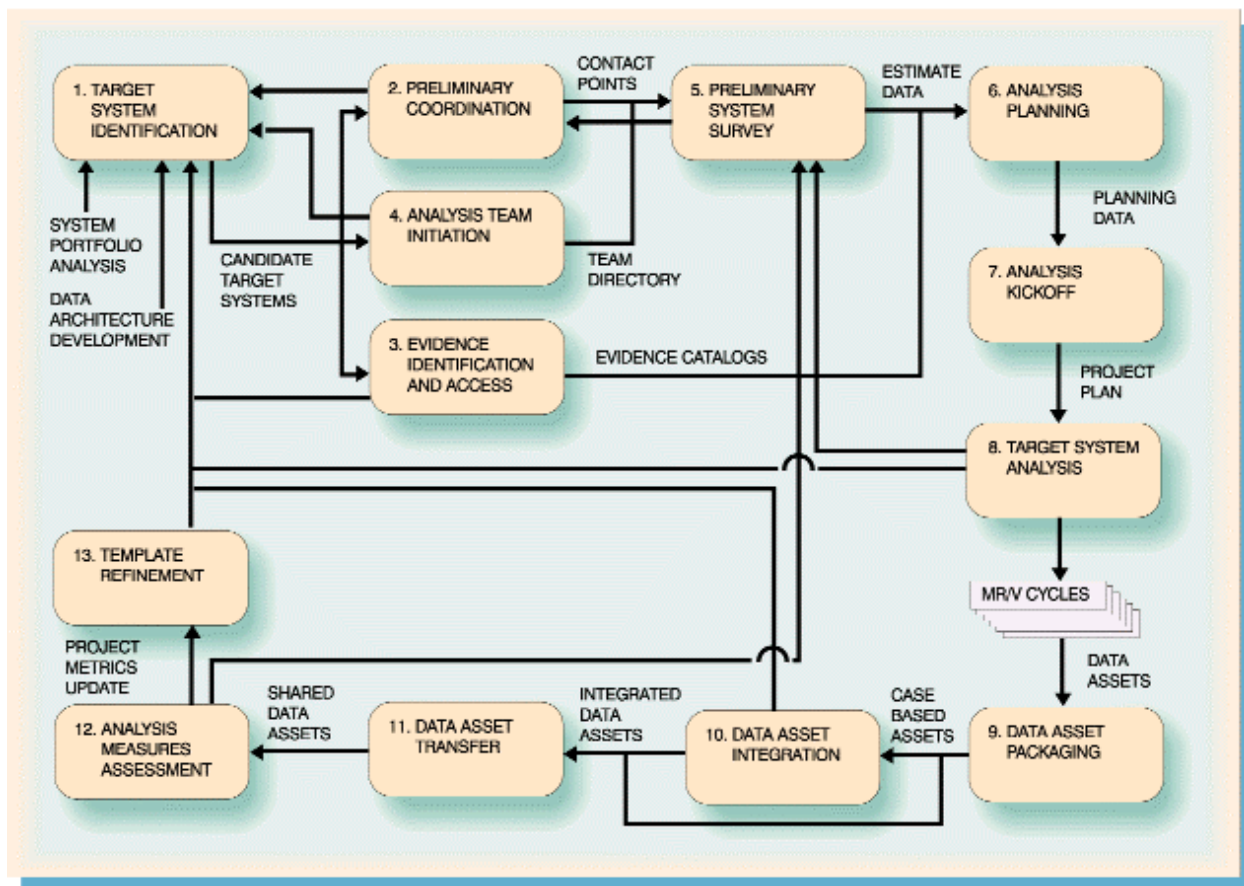
#	Activity	Outputs - Measures
<i>INITIATION PHASE</i>		
1	Target system identification	Candidate target systems - the 'size and shape' of the reengineering challenges
2	Preliminary coordination	Target system points of contact - measures of non-technological complexity
3	Evidence identification & access	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
<i>IMPLEMENTATION PHASE</i>		
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
<i>WRAP-UP PHASE</i>		
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:

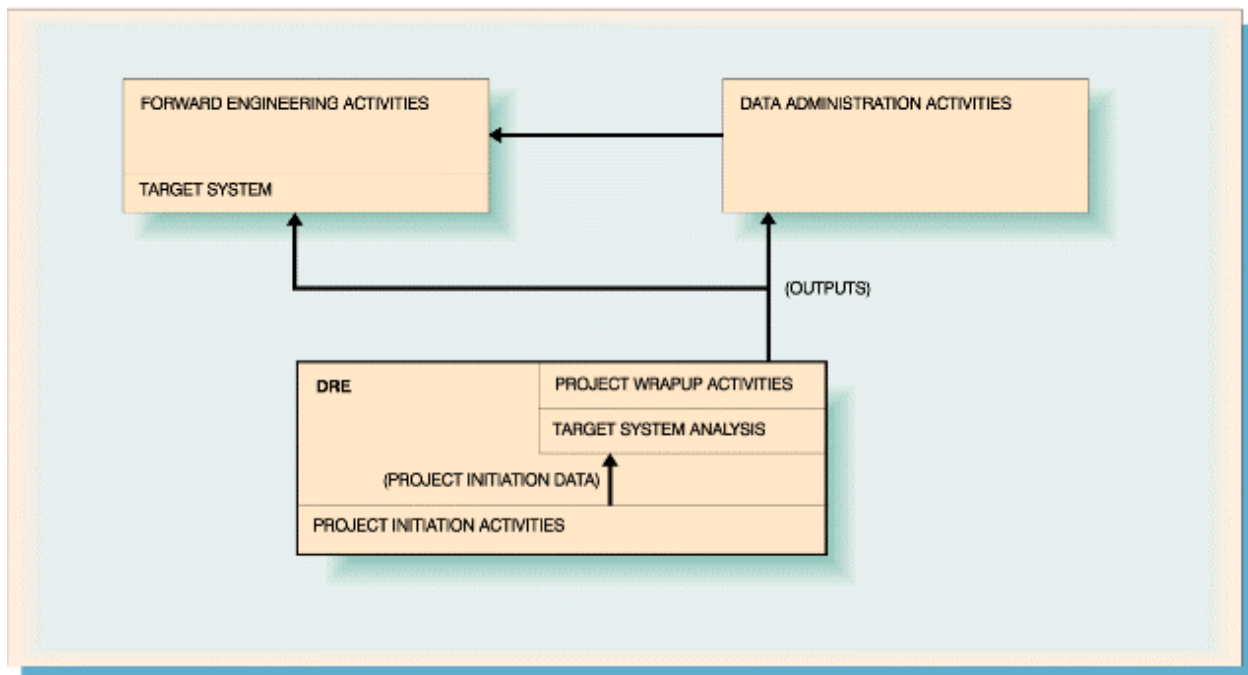
- identified and implemented solutions to required coordination issues;
- educated colleagues and project team members;
- confirmed participation commitments; and
- achieved participant consensus as to the nature of the investment in this enterprise integration activity.

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 associated 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.

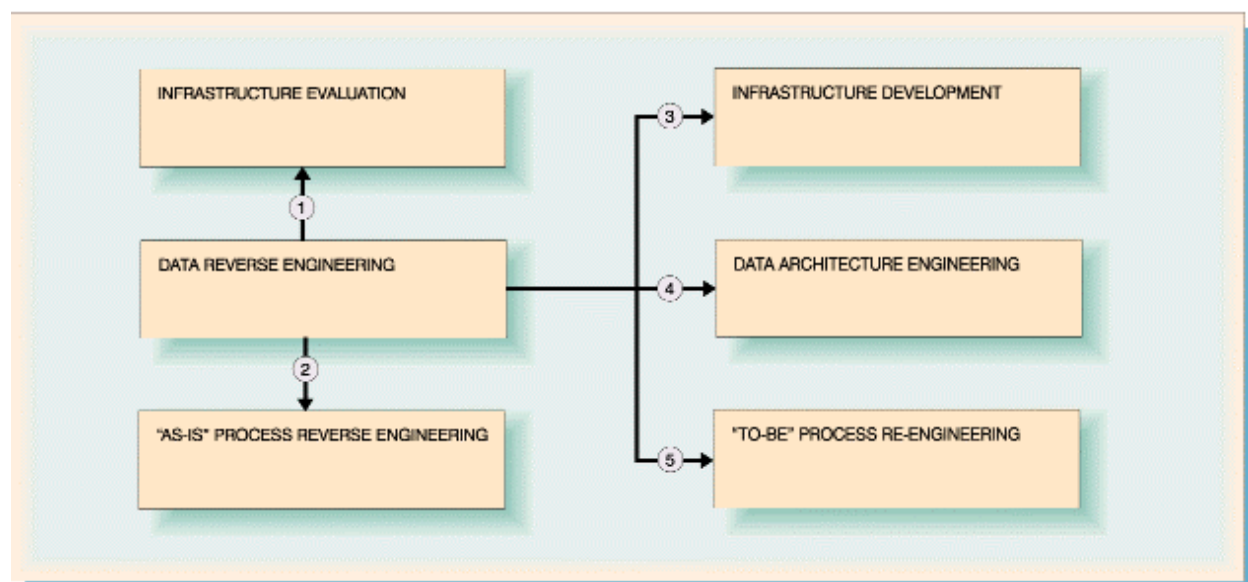
Activity 11 - Data Asset Transfer

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:

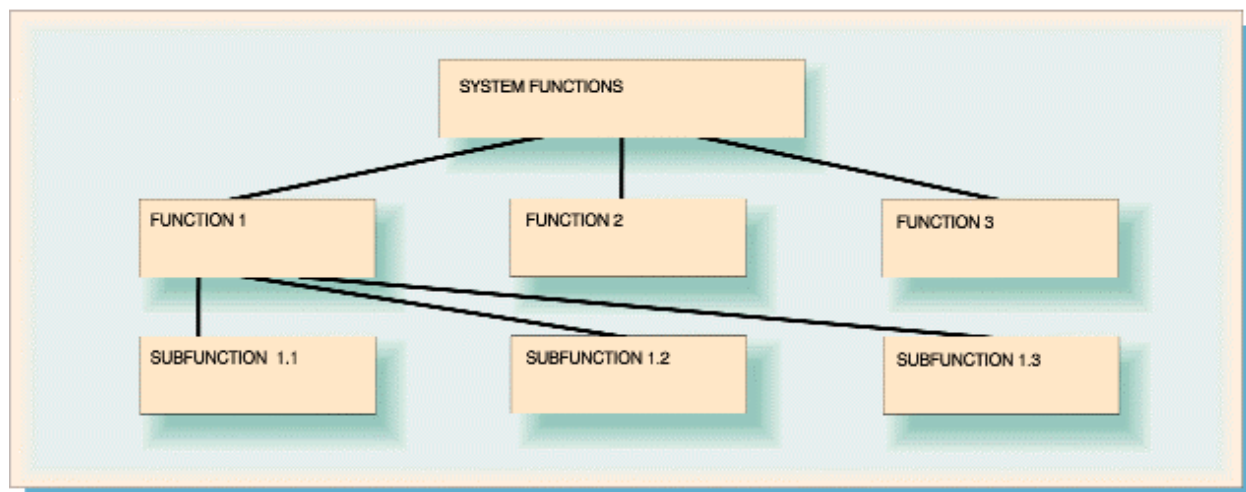
- *Leverage of data management principles* - understanding a relatively large amount of information by modeling and managing a relatively small amount of metadata. When scoping data reverse engineering projects, it is useful to understand nine possible types of data reengineering outputs (DRO) illustrated 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.

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



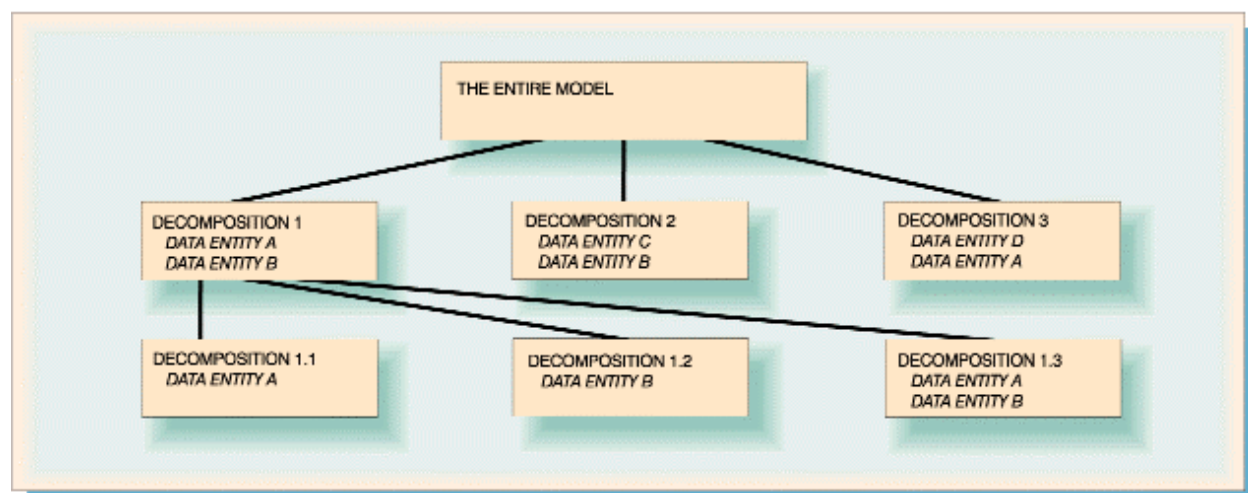
Unlike forward engineering, where analysts decomposes problems from the top-down, in reverse engineering, the decomposition is often constructed from the bottom-up by examining the system evidence. The answers are given and the question to be derived is - *what sort of functions are performed by the 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]).

<i>Implementation phase</i>			
#	Name	Activity	Output
8	<u>Target system analysis</u> (repeated until completed)		
8.1	Cycle planning	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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.

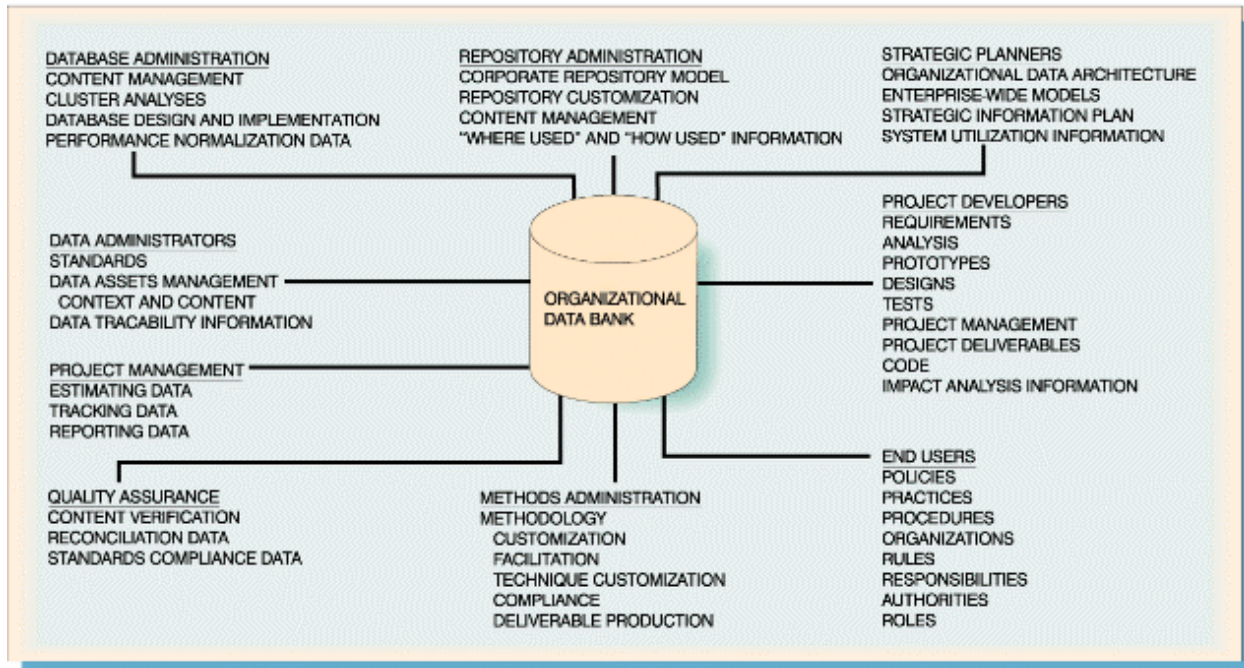
Figure 9 Possible data bank users (adapted from Selkow²¹)

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).

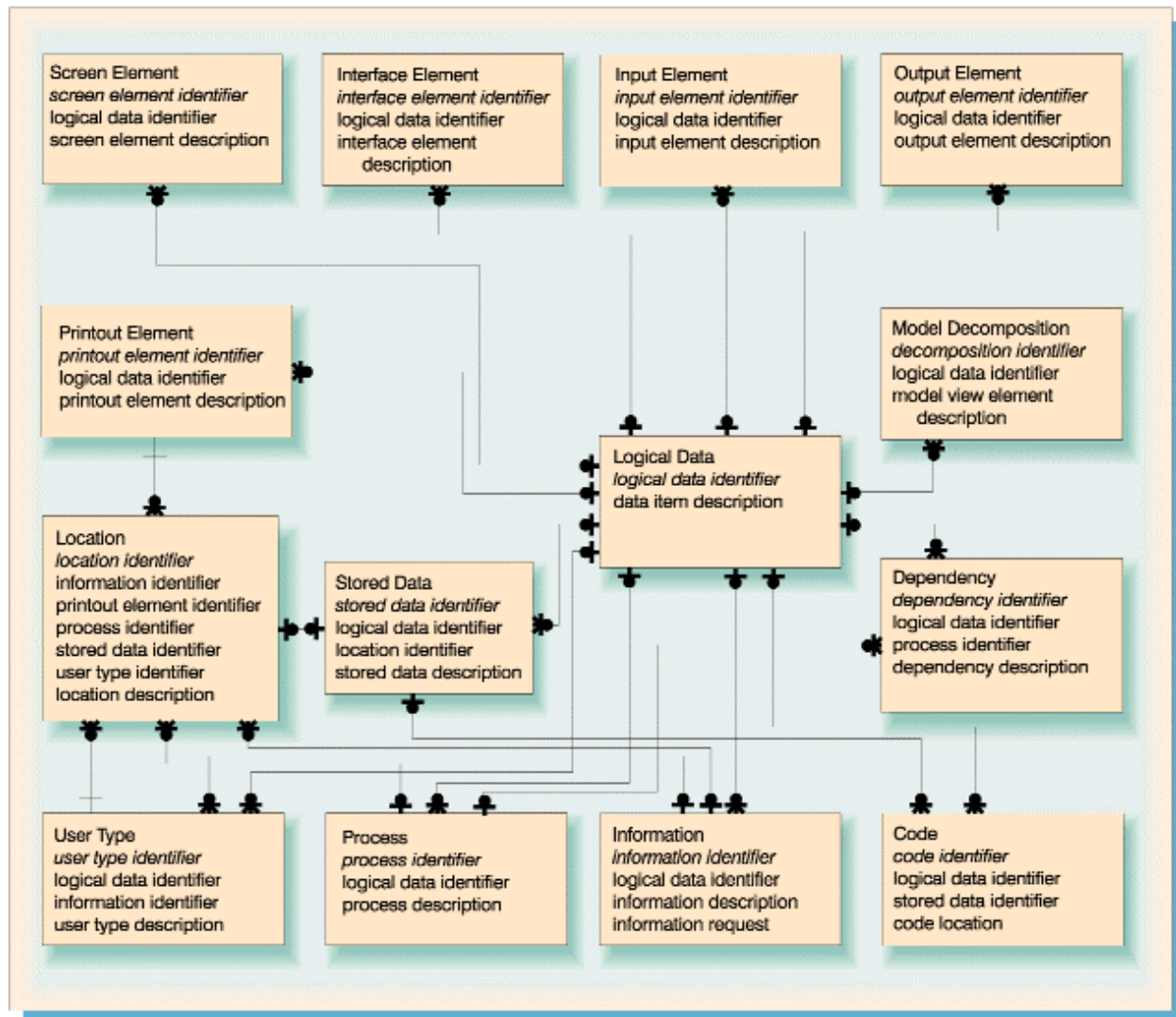
The DRE metadata model contains precise information required to understand the target system that was unavailable or disorganized before the analysis. Populating the DRE metadata model is the primary focus of target system analysis. The analysis goal is to produce validated metadata model data.

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

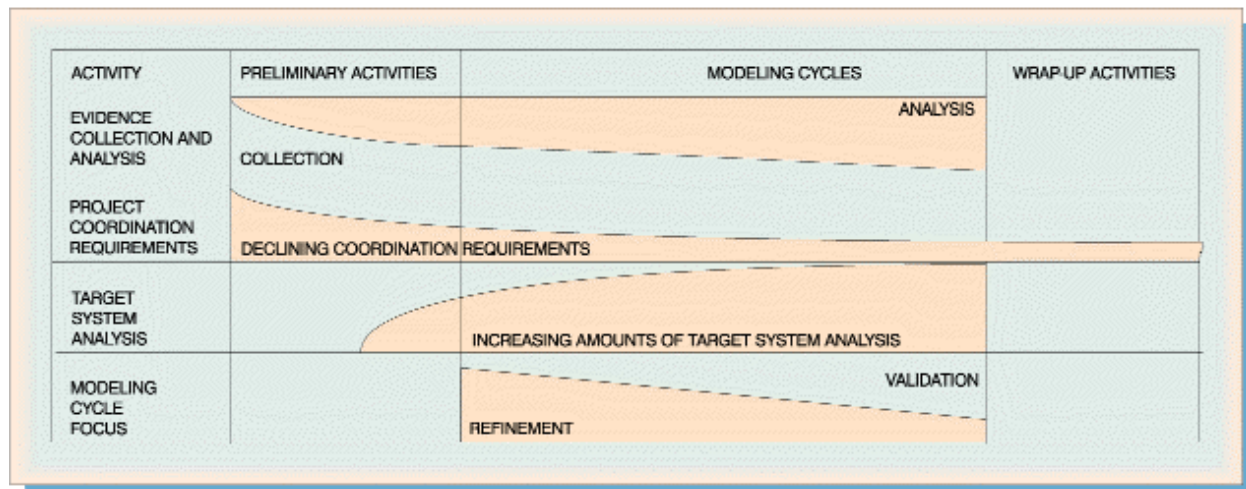
are useful individually, they are made most useful when maintained in an integrated, CASE-based organizational data bank.

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]).

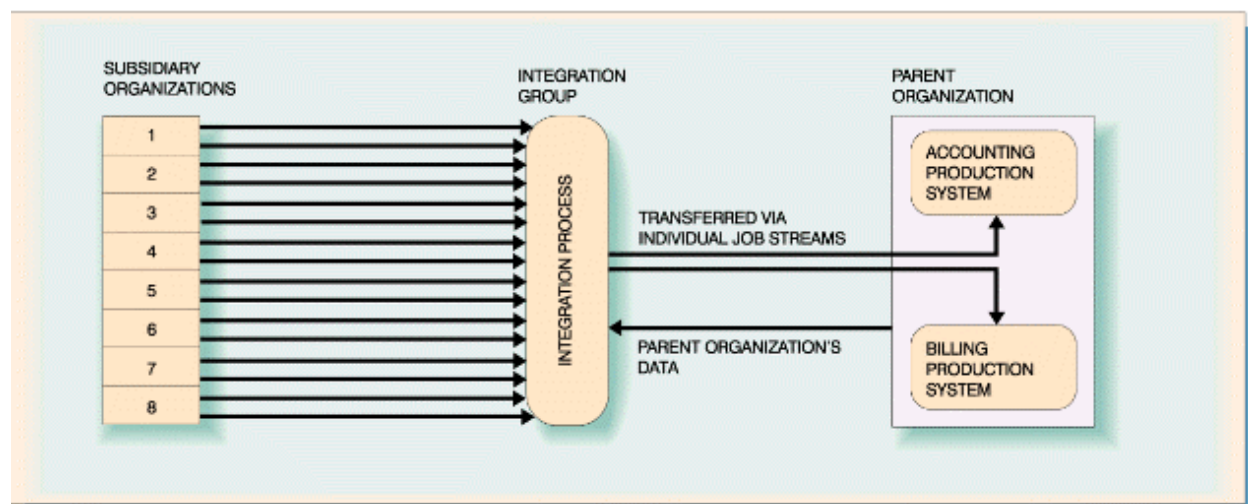
#	Primary use	Metadatum	Domain value
1	Architectural development	Data quality type	- Public - Private
2		Data usage type	- Operational - DSS I: single server node - DSS II: multiple server nodes - DSS III: distributed data and processing
3		Data residency type	- Functional - Subject - Geographic - Other
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	Likelihood that an individual data element of the total population will be accessed during a processing period
10		Data update probability	Likelihood 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

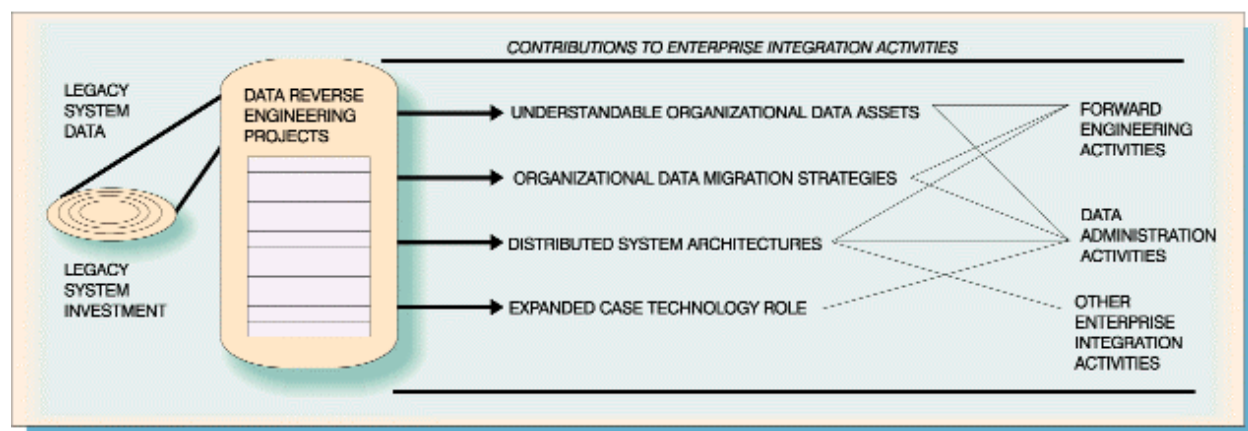
DRE has an obvious application as part of structured means of addressing organizational year two thousand (Y2K) data problems, easily providing structure for Y2K investigations. Through DRE analysis well prepares organizations to open for business on Monday - January 3rd, 2000. Target system analysis 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 advice 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 Dependencies 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).