Interoffice Memo

To: All ODOE Staff
From: Mike Kaplan, ODOE Director
Date: October 7, 2015
Subject: Database Scoping Project: Project Completion and Next Steps

---

In 2015, our agency began a project to understand how we gather, aggregate and analyze energy production and consumption data. With a long-term vision of being a central source of energy information in Oregon, we undertook a scoping effort to identify the business requirements, potential implementation methods – including business processes – and other essential information critical to achieving this vision.

Through a 2013-2015 Policy Option Package, we hired the technology consulting firm, Delaris, to lead these efforts. Over five months, their team analyzed how our agency currently collects and reports energy data and developed a plan to better integrate the various datasets into a more comprehensive system. Their work included an inventory of our current data systems, assessment of our current and future needs, and recommended pathways to developing future systems. The scoping project concluded on June 30 and was completed on time and under budget. Delaris provided a large set of deliverables that will be a valuable resources for our agency, including data models, process maps, data dictionaries, and future business requirements. They also summarized their independent assessment and findings into a final scoping report, which follows this memo.

The final report is meant to provide insight into our agency's current data systems and a variety of recommendations to begin addressing areas needing improvement. We recognize that this report is a "snapshot" and may have some information that is not entirely comprehensive due to its specific scope and duration. However, the report provides our agency with a starting point to evaluate potential improvements, and we can use the report and associated documents to inform our future data system efforts.

With the successful completion of the scoping project, our agency has an opportunity to implement some important short and long-term goals. Potential short term improvements include implementing data governance, creating a data warehouse, enhancing reporting capabilities, and improving data quality check processes. Potential long term improvements for next biennium include implementing an integrated data infrastructure and redesigning applications in use today. Other programs will have access to the information and documents from this project which may be a useful tool for refining our business processes and addressing data systems improvements.

Attached to this memo is the final report from Delaris. All other associated documentation and deliverables will be located on the MyODOE SharePoint website.

If you have any further questions or comments, please feel free to contact our Project Team: Jake Rosenbalm, Warren Cook, Michael Williams, and Kaci Radcliffe.

Oregon Department of Energy: Energy Database Scoping

By Delaris, LLC
Tony Dennis, Justin Siddon, Jeffrey Cole, and Marti Frank
June 30, 2015

1340 SW Bertha Blvd, Suite 103
Portland, Oregon, 97219
United States

Telephone (503) 445-4354
info@delaris.com
www.delaris.com

Oregon Department of Energy: Energy Database Scoping 7
Executive Summary 7
Purpose 9
Risk Findings 9
Data Management Technology is Decentralized 9
Solution: Deploy Central Data Management Technology, Migrate Data 9
Current Staffing Doesn't Meet Organizational Needs 10
Risk Mitigation: Modernize Staff Skill To Match Current Market 11
High Levels of Staff Specialization 11
Mitigation Strategy: Standardize Training, Centralize Systems 12
Data Standards Aren't Enforced at Entry 12
Mitigation Strategy: Implement Automated Quality Controls 13
Inconsistent Data Policies 13
Mitigation Strategy: Adopt a Data Governance Policy 14
Solutions 15
Customer Relationship Management 15
Recommendation: Integrate CRM Software With Energy Tracking System 16
Online Transaction Processing System 16
Discussion 16
Two-Tier Architecture 16
Service Oriented Architecture 17
Recommendation: Implement OLTP System With SOA 17
Report Server, Data Warehouse 17
Discussion 17
Server Load 17
Understanding The Data Structure 18
Recommendation 18
Reporting server 18
Tactical Reports 19
Operational Reports 19
Strategic Reports 19
Recommendation 19
Data Governance Policy 20
Delaris Recommend ODOE Staffing Structure 21
Recommendation 22
Conceptual Data Design 23
Outline Data Model 23
Systems Design 24
EAV Design 25
Data View 26
Recommendation 27
Simple vs. Integrated Measures 27
Recommendation 29
Implementation 30

Phase 0: Present State 30
Risks Mitigated 31
Risks Remaining: 31
Phase 1: Implement Report Server 31
Risks Remaining 32
Phase 2: Centralize Data Entry 33
Risks Mitigated 34
Risks Remaining 34
Phase 3: Integrate Remaining Programs 35
Risks Mitigated 35
Risks Remaining 36
Phase 4: Integrate Utility and Transportation Data 36
Risks Mitigated 37
Risks Remaining 37
Architecture choices 37
Commercial Software Packages to Manage Energy Incentive Programs 37
Sources of Failure 37
Recommendation 38
Closed vs. Open Source Tools 38
Toolsets 38
Commercial Products 38
Recommendation 39
Full Relational vs. EAV Data Models 39
EAV Advantages 39
EAV Disadvantages 39
Recommendation 40
Relational vs. Non-Relational Databases 40
Relational Databases 40
Non-Relational Databases 40
Recommendation 40
Contracting Strategies 41
Multiple vs. Single Contract Strategy 41
Recommendation 42
Fixed-Price vs. Cost-Plus Contracting 42
Recommendation 43
Multiple Small Projects vs. Single Big Project 44
Recommendation 44
Cost 45
Buy vs. Buy and Modify vs. Contract Build vs. Internal Build 45
Buy and Modify (best odds of success) 45
Contract Build (diminishing odds of success) 45
Internal Build (least desirable) 45
Budget estimates 46
Buy - $300K to $500K 46
Buy and Modify - $1.3 million to $1.5 million 46
Contract Build - $1.5 to $2.5 million 46

Internal Build - $1.5 million 46
Possible Synergies With Other Organizations 47
Energy Trust of Oregon (ETO) 47
Northwest Energy Efficiency Alliance (NEEA) 47
Methodology 48
Phase Descriptions 48
Kickoff 48
Discovery 48
Pre-Interview Survey 48
Interview 48
Deliverables: 49
Validation and Metadata 49
Deliverables: 50
Validation and Risk 50
Deliverables: 50
Document 50
Recommend 50
Deliverables 50

Oregon Department of Energy: Energy Database Scoping

Executive Summary

The Oregon Department of Energy's (ODOE) primary mission is to promote a safe, clean, sustainable energy future for Oregon. ODOE accomplishes this through administration of energy savings programs and by serving as a primary source of energy sector data for policymakers and constituents at large; Delaris was hired by ODOE to assess its current ability to continue these services. Over the course of the last six months, Delaris personnel have directed extensive discovery of ODOE's current infrastructures, policies, and procedures. Delaris believes current staffing, policies, and data systems are endangering ODOE's ability to continue fulfilling its mission.

Primary Risks

1. Data management technology is decentralized – programs record data in distinctly separate stores, relying on reporting processes to combine data across programs
2. Current staffing doesn't meet organizational needs – IT is grossly understaffed, and current personnel don't possess necessary skill to implement future systems
3. High levels of staff specialization – staff fulfilling similar roles in different programs can't support one another, key staff are overly relied upon to compensate for limited technology
4. Data standards aren't enforced at entry – Inconsistently implemented data entry checks are primarily human reliant, with almost no automated data quality checks completed
5. Inconsistent data policies – no organization wide data policies have been emplaced, programs don't operate on a consistent definition of data

Recommendation

Delaris recommends the State of Oregon invest in a major upgrade of ODOE data systems to centralize data storage, definition, and standards. A central data management system should:

- Serve as a central data store for all program data
- Be based on supportable, industry standard database technology
- Feature a service-oriented architecture
- Provide users a web front-end for data entry and referencing
- Separate data servers for data entry (On Line Transaction Processing), and reporting (data warehousing)
- Integrate with commercial off the shelf support services for specific data management

Delaris discovered existing systems that should be further investigated for licensing opportunities (Energy Trust of Oregon, US DOE, etc.) as their current offerings closely align with ODOE's needs.

Evaluation of large Enterprise Resource Planning platform vendor (e.g., SAP, Microsoft Dynamics) offerings leads us to believe the unique data, workflow, and reporting requirements within the energy program industry would be prohibitively complicated, expensive, and brittle to maintain if built within Enterprise Resource Planning software.

Implementation of a new data management system should be completed in discrete phases by external contract resources. Exclusive use of internal resources will place significant risk on the project as current staffing is stressed with maintaining operational continuity, severely limiting available development time. In parallel with systems investments, Delaris recommends expanding internal programing and maintenance IT personnel.

Based on an a "buy and modify" of the available solutions from specialty vendors, Delaris estimates licensing a product would require a $300,000 - $500,000 fee plus a three member project team for eighteen months, increasing the total project cost to ~$1,500,000.

Additional discoveries and recommendations are included in the following report.

Purpose

The Oregon Department of Energy's (ODOE) primary mission is to promote a safe, clean, sustainable energy future for Oregon. ODOE accomplishes this through administration of energy savings programs and by serving as a single source of energy sector data for policymakers and constituents at large; Delaris was hired by ODOE to assess its current ability to continue these services. Over the course of the last six months, Delaris personnel have directed extensive discovery of ODOE's current infrastructures, policies, and procedures. Delaris believes current staffing, policies, and data systems are endangering ODOE's ability to continue fulfilling its mission. Delaris's tasks included, but were not limited to:

- Identifying energy data sources
- Identifying potential database users
- Identifying analytical needs
- Identifying reporting needs
- Identifying technical alternatives to meet Agency's goal of creating a central source of energy information in Oregon, and
- Providing recommendations and estimated costs for paths to implementation

Risk Findings

Data Management Technology is Decentralized

By culture and design, many programs within ODOE had a short-term life expectancy, which reflects in a short-term view of program data and its utility. In practice however, programs are extended and in some cases persist for decades. As a result, IT systems to support programs were also built for short term use and consistently prove inadequate over the extended program lifecycles

Fragmented Systems

The design and execution of programs, as stand-alone entities, means systems don't fit together succinctly for reporting or portfolio management. Extensive staff time is required to combine data from multiple systems.

Difficulty Locating Data

Internal historical data searches require locating program staff responsible for the data in question, often resulting in the reference of a "shadow system" (i.e., stashed spreadsheet or document) inaccessible by their peers to provide the answer. While similar problems continue to persist inside businesses and government, Delaris feels ODOE is an extreme case.

Solution: Deploy Central Data Management Technology, Migrate Data

Centralized data systems reduce the need for "shadow systems" to keep track of details that are important to programs, particularly if the system is flexible and responsive enough to quickly add attributes when needs change. Centralized data

systems also force the recognition of savings allocation between programs at the project level, which provides a more accurate view of the program when viewed independently.

Various enterprise and open source solutions fit ODOE's requirements. Delaris recommends ODOE carefully assess available technology with a long-term (10 year) view of developer availability.

Post implementation, historical program data should be migrated into reporting systems so legacy data can be referenced from central tools.

ODOE needs to procure a system that is flexible enough to adapt to the unique needs of current and future programs. Equally important is increased oversight into the program proposal cycle, so data processing and storage requirements are incorporated into the full program plan. Where possible, common fields and data definitions need to be implemented, to prevent unnecessary duplication of effort.

Current Staffing Doesn't Meet Organizational Needs

ODOE systems need to be usable and supportable by internal staff. ODOE doesn't currently possess staff with experience or training with modern data management tools.

Legacy Technology

Large amounts of data are still processed and stored using Visual FoxPro[1]. Visual Fox Pro's development ceased in 2007, and its originating product, FoxPro's final version, was released in 1994. While Visual FoxPro does support relationships between tables, it is *not* considered a relational database management system (RDMS) because of its lack of transaction support.

While programs written in FoxPro will continue to run for the foreseeable future, it is considered an obsolete technology. Better solutions have emerged, most notably an entire generation of relational databases (e.g., MS SQL, Oracle, MySQL) providing richer features and better development tools. Consequently, few developers are still working on FoxPro projects, which means locating qualified personnel will increase in difficulty with time.

ODOE currently has a single FoxPro developer. In addition to updating the FoxPro code base, he supports changes to existing programs within other technology stacks, constituting himself as a single point of failure. His departure would also result in much of his institutional knowledge, bound into the FoxPro code, leaving with him. It would be possible for ODOE to find and train developers to program legacy systems, but it would be inefficient, time consuming, and costly. Delaris' evaluation of the FoxPro systems is there is not enough legacy knowledge bound into the software to warrant the additional efforts required to maintain the systems.

[1] http://en.wikipedia.org/wiki/FoxPro

Data reporting tools are also obsolete. Reports are written in ReportWorks, an old, non-standard tool. Not enough ODOE staff possesses training in modern reporting tools (e.g., Crystal Reports, MS SSRS, Oracle Reports, etc.)[2].

Risk Mitigation: Modernize Staff Skill To Match Current Market

Train existing employees in modern technologies. Provide staff the tools and latitude to explore and sample technology as markets develop. Recruit new staff with long-term technical horizons in mind. Delaris believes significant technological shifts are beginning.

Changes In Technological Landscape

Delaris believes long-term developer market for programmers with Enterprise software experience (.NET and Java, in particular) is beginning to decline.[3] Delaris recommends caution with quick decisions to classic enterprise software packages. To quote: [4]

"There are plenty of companies out there still looking for good .NET developers. They will be in demand for a long time; like COBOL developers. Believe me, you don't "just rewrite" 4 decades of COBOL; the same holds true for the .NET stack. However, I did draw a conclusion that the .NET stack is on its way out. There are a number of reasons we believe it's on its way out:

1. *The stack is heavily server-side dependent.*
2. *The server-side technology is very heavy and relies on an even clunkier web server for its exposure. Even the lightest server-side technology, ASP.NET Web API, is still heavier than NodeJS.*
3. *The market has grown weary of the high cost of entry due to licensing fees. To get into Visual Studio Ultimate (necessary for any very large projects) you need to invest $13K; that's insane!*
4. *The typical .NET stack is not homogeneous in nature. The data alone, and the transitions it must go through from database engine, to data model, to view model, to JSON, and back again tells the tale. It doesn't matter how much syntactic sugar you wrap it in. It doesn't matter how many frameworks. The stack is not homogeneous. "*

The implication for ODOE is developers with the appropriate skills may be plentiful in the short term, longer term recruiting efforts may be more difficult potentially placing ODOE in the same position in five years as it is in today.

High Levels of Staff Specialization

A pervasive culture of specialization exists within ODOE. Many employees have been with ODOE for a long time serving the same role. While attentive to individual

[2] http://en.wikipedia.org/wiki/List_of_reporting_software
[3] http://blog.jonathanoliver.com/why-i-left-dot-net/
[4] https://www.airpair.com/mean-stack/posts/developers-moving-dotnet-to-mean

program needs, this specialization has created unsystematic work processes and concentrated critical organization knowledge to select staff.

Unsystematic Processes

Records keeping is based in spreadsheets or obsolete database applications, placing the entirety of data quality procedures on the individual. Because of this, vast discrepancies exist between otherwise similar processes by program. Individual program expertise must be cultivated within each employee requiring an undue training burden to hire new employees.

Associated problems are particularly apparent during reporting. Currently, one staff member completes the *"Annual Energy Savings Report"*, a critical data source for multiple executive and legislative level reports. Complex processes must be completed to normalize data across programs. Efforts to document this process have been made but ability to transition this process to new staff remains minimal. Problems are further exacerbated by his current part-time employment with the agency following his retirement five years ago. His departure would pose a significant challenge to the agency's continued function.

Many Single Points of Failure

Systems maintenance and support is distributed to individual resources within programs. No formal policies and procedures are enforced during staff departure. For example, the Small Energy Loan Program (SELP) program utilizes software that was specified and maintained by a previous employee whom is still contacted for specific software assistance. In another case, Residential Energy Tax Credit (RETC) program analysts are unable to answer program process questions without program manager assistance.

Mitigation Strategy: Standardize Training, Centralize Systems

Document standard position onboarding procedures and deliver continued cross-function training to all staff. Standard training ensures all staff can function according to agency role definitions while cross- function training allows staff to fill in for other team members if needed. Currently, ODOE's ability to meet business obligations during the absence of staff is minimal.

Centralizing data models and databases simplifies business processes. Staff within programs can fluently transition between programs. In addition, centralizing systems results in data unification, reducing dependency on manual reporting functions.

Data Standards Aren't Enforced at Entry

Few ODOE programs have modern, form-based data entry tools and as a result data quality suffers. Primary quality control (QC) is relegated to post-entry reporting processes. Associated problems are compounded by over-reliance on individual staff program knowledge or simple data checks (e.g., sums and counts). When errors

are discovered, current correction processes are amorphous; there are no systems in place to confirm errors are fixed in source data.

Of particular concern is data quantity variation due to miskeyed entry. Errors of this type tend to increase summed report values, for example, consider two errors; an extra digit entered into a savings value, and a missed digit in another. If the assumed correct savings value is ~234 kWh, the addition or subtraction of a digit skews the value by an order of magnitude in either direction.

Record #	Correctly Keyed	Offsetting keying errors
1	234	2334 (one digit double-keyed)
2	235	235
3	236	26 (one digit missed)
Sum	705	2,595

Current process to check these errors are visual, reporting staff looks for differences by numerical columns, if digits don't line up, the entry gets rechecked. This technique is unreliable and irreproducible.

Mitigation Strategy: Implement Automated Quality Controls

Automation enforces data quality rules at input, preventing bad data from entering systems. In addition, instituting a system of check-reports will ensure errors are identified and corrected early, minimizing cascading effects bad data introduces and streamlining corrections processes while source data is easier to access. Delaris recommends implementing two automated QC checks.

QC 1 - Input Validation

Implement simple type checking validation to data entry; numeric validation (e.g., 12# instead of 123) and range checking (e.g., 234 instead of 2334 kWh).

QC 2 - Quality Reports

Construct daily data QC reports. Summarizing the day's data (summing, averaging and calculating standard deviation among values) frequently ensures staff attention is directed to erroneous data while the source data (applications) are nearby and fresh in memory.

Inconsistent Data Policies

At a high-level, programs are run with a similar workflow, applications are received, data is recorded, business rules verified and an incentive is dispersed. At a lower level, programs diverge; staffing, energy savings measures, and legislative directives all vary between programs. Chasms are compounded by program independent IT contracting. The combination of factors has resulted in divergence in data processes, and as a result data policies are difficult to declare and maintain.

Inconsistent Data Definitions

Data is defined differently between programs. One program my interpret data in ways that may skew data when compared to another. ODOE relies on reporting staff to accurately convert programs to data formats that fit one another.

Dual Entry

Data entry processes across programs require staff to enter application data into one, two, or in some cases three different systems. Through our analysis we discovered multiple causes of dual entry including:

- Systems lack of program performance monitoring tools
- Poor user interfaces
- Incomplete system development
- Insufficient staff training

Dual entry processes are an inefficient use of staff time and introduce additional complexity associated with "dual master" records.

Multi-Program Application

Applicants may apply to multiple programs at a time, because of the siloed program structure and lack of standard review processes, a single project will be entered may be counted more than once. Currently, duplicate savings claims are retroactively resolved between programs during reporting by calculating an annual percent savings overlap factor. While effective, calculated overlap factors are never applied to program source data, resulting in inaccurate single program data if removed from context.

Mitigation Strategy: Adopt a Data Governance Policy

Data governance policies will solve these problems by centralizing the program, system, and data definition process, taking a systemic, long-term view of the program portfolio lifecycle.

Data-governance Council Functions

1. Create a data governance council chaired by a senior-level manager with appropriate cross-program decision-making authority.
2. Recruit additional data governance council members from across the organization, including; operations, IT, planning, and management. (See the section later in the report regarding Integrated Product Teams.)
3. Propose new program data needs to council. The council governs allocation of resources (IT work hours or contracting dollars) dedicated to bringing a program on-line or changing existing programs.
4. Define consistent data definitions applicable across ODOE through the data governance council.
5. Control central data system operational features (screens, tables, reports, or other) and configurations through data governance council.

Formal data governance practices as discussed, ensure deliberate decisions with regards to system change, while maintaining consistent processes and data definitions throughout the organization.

Solutions

Customer Relationship Management

A Customer Relationship Management (CRM) application is a standalone system used to track and maintain a comprehensive record of customer communications. Data improves customer service, reporting, and serves as a data source for follow-on communication and marketing efforts. CRMs are generally integrated with other communications clients like phone and email to provide a customer-centric view of the world. An example of a CRM integrating with ODOE processes if found below.

Case Study: CRM integrated with ODOE processes

ODOE's CRM has been configured and integrated. Now, when ODOE receives an application for an incentive, applicant contact data entered in the Energy Tracking System would check with the CRM to see if a contact already exists for that customer. If a record was found, staff is notified, and previous customer interactions become instantly available. ODOE staff can now select the correct address if the addresses match, or correct it in the CRM if it doesn't. A record of the application is then stored so that it becomes part of the customer's record, including the date the application was received. If a customer calls in asking for the status of one or more applications, the status of each will be readily available. Modern CRM's can track multiple addresses per customer and can associate multiple customers with sites, particularly useful for large customers.

CRM's have become a well-established part of business IT, their maturity means they should be procured off-the-shelf, and integrated into the energy tracking system. Delaris recommends ODOE pursue an in depth analysis to determine which CRM package will best fit ODOE's needs.

There are several major CRM's, with the market leaders shown in the following table[5]:

Vendor	2013 Revenue	2013 Share (%)	2012 Revenue	2012 Share (%)	2008 Revenue	2008 Share (%)	2007 Revenue	2007 Share (%)	2006 Revenue	2006 Share (%)
Salesforce.com CRM	3,292	16.1	2,525.6	14.0	965	10.6	676.5	8.3	451.7	6.9
SAP AG	2,622	12.8	2,327.1	12.9	2,055	22.5	2,050.8	25.3	1,681.7	25.6
Oracle	2,097	10.2	2,015.2	11.1	1,475	16.1	1,319.8	16.3	1,016.8	15.5
Microsoft Dynamics CRM	1,392	6.8	1,135.3	6.3	581	6.4	332.1	4.1	176.1	2.7

[5] http://en.wikipedia.org/wiki/Customer_relationship_management#Market_leaders

Others	11,076	54.1	10,086.8	55.7	3,620	39.6	3,289.1	40.6	2,881.6	43.8
Total	20,476	100	18,090	100	9,147	100	7,674	100	6,214	100

Salesforce.com

Salesforce.com is the current CRM market leader. Their solution is provided in a Software as a Service (SaaS) model. Their extensive interface allows organizations to integrate a variety of back-office systems into the Salesforce view. The disadvantage of Salesforce is that it may not provide enough on-premise control for some organizations including ODOE.

SAP and Oracle

SAP and Oracle have both been losing ground in the CRM market and are probably more suitable for organizations that have established SAP or Oracle solutions.

Microsoft Dynamics CRM

Microsoft Dynamics CRM' primary attraction is its seamless integration with the Microsoft Office suite, and particularly the Outlook email client. Microsoft Dynamics CRM is offered in both a Software-as-a-Service and on-premise licensing models.

Recommendation: Integrate CRM Software With Energy Tracking System

ODOE to survey the market for appropriate CRM software, and then integrate it into a larger energy tracking solution. Delaris recommends ODOE also license and implement Microsoft Dynamics CRM to support it was built to support this integration.

Online Transaction Processing System

Online Transaction Processing (OLTP) systems facilitate and manage data entry and retrieval processes.

Discussion

The central OLTP system provides day-to-day transactional support for ODOE's primary business. Delaris recommends ODOE implement a solution similar to others available from specialty vendors. Some examples of these systems utilize a MS-SQL server as its back-end, which also includes SQL Server Reporting Services (SSRS), SQL Server Integration Services (SSSIS), SQL Server Analysis Services (SSAS), database replication, and others. This toolset fits well with ODOE's current Microsoft-centric environment.

Two-Tier Architecture

Two-tier architectures or client/server architectures, consist of a client application containing both the display and business rule code, communicating over a network with a database server whose responsibility is to record the results. While simple, the tightly coupled[6] nature of client/server systems lacks the flexibility and scalability found in more modern approaches.

[6] http://www.informit.com/articles/article.aspx?p=349749&seqNum=5

Service Oriented Architecture

A SOA based architecture "loosens" the coupling between the applications layers. Most commonly this structure is implemented by inserting a generalized service layer called an Application Programming Interface (API). The API service layer incorporates most of the data and business logic checks traditionally performed by the client, but is generalized so the same code can serve multiple purposes and interfaces. This makes the application more flexible as programs and their associated structures are changed.

Recommendation: Implement OLTP System With SOA

Delaris recommends ODOE implement a service-oriented architecture (SOA)[7] (three or n-application tiers) [8] instead of a more traditional two-tier[9] architecture. While cheaper to build, two-tier applications are more difficult and expensive to maintain while the thin footprint SOA clients contain, means they cost less to maintain, customize and interface with legacy systems and programs.

Use of a centralized OLTP system will allow ODOE to consolidate data models into a central data view enabling cross-program data quality rules, definitions, and accounting standards to be enforced uniformly. It also consolidates data into a central repository for reporting. The OLTP system should interface with the CRM system to track relations between customers and sites, while the OLTP system tracks relationships between programs, projects, tax credits, and savings.

Report Server, Data Warehouse

Discussion

In addition to an OLTP system Delaris recommends ODOE implement a separate reporting server and data warehouse, as reporting directly from the OLTP servers presents some challenges.

Server Load

OLTP transactions tend to be small, short, and fast. For example, when a staff person wants to look at a customer record, they receive the first few "detail" records (typically a single page), which are efficiently located using an index. Writes are one record long, sums and counts are written into the database and are updated when the base record changes instead of being recalculated every time. OLTP system load tends to be smooth as users naturally spread their requests with the natural randomness of doing work; loads grow and decline based on the time of day, month of the year, etc. The smooth load curve a normalized, indexed relational data model

[7] http://en.wikipedia.org/wiki/Service-oriented_architecture
[8] http://en.wikipedia.org/wiki/Multitier_architecture
[9] http://en.wikipedia.org/wiki/Client–server_model

presents is adequate for this type of application. Reporting presents a different picture.

During report execution many tables are joined to form a larger whole and table scans (examining every record in a table) happen more often because there are more aggregate functions (e.g., sums and counts). Often odd or "expensive" table joins are needed to categorize the data in different ways. Reporting server load tends to radically shift. Resource spikes can happen unexpectedly, slowing OLTP transactions down (transactions normally taking seconds may take minutes while reporting users may not observe any performance change).

Because of the differences in load patterns, it's important to separate the tasks supporting systems. Classically, OLTP databases and reporting systems run on separate hardware, shielding traffic from one-another. Data in data warehouse systems are then updated through data synchronization methods (Extract-Transform-Load (ETL).

Understanding The Data Structure

Most users think of data as rows in a spreadsheet. Relational models used in OLTP databases can be confusing to users, so primary data analysis is commonly completed in Excel.

Because of this, many enterprise systems incorporate a reporting server that separates transactional data from reporting data. In practice, transactional data is summarized into a "flattened" reporting structure at a regular interval (e.g., once every twenty four hours) and the resulting data is made available in a "data mart" for users to analyze with their choice of tools.

There are many different schemes used to populate reporting services, but most involve some sort of standard ETL[10] process. In Microsoft-centric systems this is often accomplished using SSIS or other integration services, although each situation will vary.

Recommendation

Implement a data warehousing system that provides separate OLTP and reporting facilities.

Reporting server

ODOE should implement multiple levels of reports; tactical, operational, and strategic.

[10] http://en.wikipedia.org/wiki/Extract,_transform,_load

Tactical Reports

Standard users use tactical reports regularly, typically "canned" or pre-formatted they serve as immediate feedback for day-to-day business processes (e.g., checking the sum of data entries for a particular day).

Operational Reports

Operational reports are aimed at a tier higher than tactical and typically serve two purposes; identify systemic issues and explore solutions. An example of a report used to identify a systemic issue might be one that compares the current quarterly performance to previous year's quarterly performance over a variety of categories (tax credits, number of projects, number of applications, measure type, sector, etc.). Deviations would prompt the manager to use the second category of operational report to "drill down" into the data set to try to identify root causes. Operational reports typically deal with longer time horizons, more dimensions or factors, and are more difficult to read and interpret. They feature more customizations (e.g., filters, grouping, processing, etc.) and are harder to build and troubleshoot. These reports requirement more effort to build because of the amount of data linking between data and projects.

Strategic Reports

Strategic reports are designed to inform the highest levels of decision-making. Typically involving review of long timelines (years or decades), strategic reports help identify deep trends. Reports may be canned, but generally have more customization available than operational reports. These reports identify deep trends, anticipate change, and to inform the future of the organization. This typically means that strategic reports are more ad hoc, requiring a higher level of analyst to build and test, and take more resources to run. They can also be characterized by the use of deeper statistical techniques and try to uncover hidden meaning within large data sets ("deep data" analysis).

Recommendation

Develop a suite of standardized reporting tools to meet the varying needs of tactical, operational, and strategic reporting. Train users on the operation of reports.

Explore solutions like Crystal Reports and MS SSRS to create pre-programmed tactical reports easily accessed from user desktops.

Review more sophisticated tools for operational reporting on a case-by-case basis. Microsoft has a suite of data-cube services that allow data to be analyzed from multiple dimensions. Excel is also a good tool at this level, but additional training is required to get full performance from the tool.

Begin strategic reports with Excel, and transition to more robust statistical analysis packages like SAS[11] or SPSS[12].

[11] http://www.sas.com/en_us/home.html

Continually test reports and their configurations. Implement systematic report change processes. Standard report modification procedures should be followed and changes should be documented like other software.

Data Governance Policy

Delaris recommends ODOE establish formal data governance policies prior to systems implementation.

1. **Establish project championship within organization**
   IT should not lead this project. This project touches all aspects of the business, and it therefore should have a business champion.

   > ***Business champion[13]:*** *A business champion is a person who voluntarily takes extraordinary interest in the adoption, implementation, and success of a cause, policy, program, project, or product. He or she will typically try to force the idea through entrenched internal resistance to change, and will evangelize it throughout the organization. Also called change advocate, change agent, or idea champion.*

   The business champion is responsible for overcoming obstacles, both internal to ODOE (culture, time management) and external (budget shortfalls, manning gaps, regulatory hurdles). The diverse functions a business champion must serve means this role should be filled by a senior manager with authority to influence resources to accomplish the mission.

2. **Get high-level buy-off from management**
   Fully understanding of the amount of changes new data systems cause in an organization at the senior-level is critical. It is imperative that the risks and benefits be discussed and understood up-front by the management team with process buy-in.

3. **Establish Data Governance Policies**
   Procedures need to be established to govern data. A Data Governance committee should be created that incorporates business management, program line staff, IT support staff, and developers to discuss how data should be standardized across programs while continuing to fulfill the unique needs of each individual program.

4. **Clean forms and business processes**

---

[12] http://www-01.ibm.com/software/analytics/spss/
[13] http://www.businessdictionary.com/definition/champion.html

Critical review of business processes within programs is needed. Remove unnecessary procedures, data, and wasted motion (i.e., double entry). Carrying extraneous fields forward will cost more money and increase the projects complexity.

Questions about data:

- *Is this field necessary to administer the program?*
- *Is this field necessary to measure or evaluate the program?*
- *Is this field necessary to measure or evaluate the agency?*
- *Does this field fulfill a regulatory requirement?*
- *Have there been enquiries from outside ODOE to report on this field?*

5. **Develop internal talent**

Develop the right skills in internal technical resources; continued education, budget for conferences, and purchase training materials. Encourage keeping abreast of changes in the industry. Select modern technologies and use them.

Increase ODOEs IT staff, short-staffed. ODOE is a data-centric organization and as a result, should have a heavier IT staff than other comparable state agencies. Delaris' review of other energy efficient industry partners, with a similar size and mission to ODOE, has an IT staff varying between seven and seventeen depending on the development and testing load. In addition, extensive use of external contract resources to fill short-term gaps is also common.

Delaris Recommend ODOE Staffing Structure

Development/Implementation Structure

i. IT Director
    1. IT Desktop and Server Manager
        a. Database Administrator (DBA)
        b. Desktop Support Technician, Grade 1
        c. Desktop Support Technician, Grade 2
        d. CRM Administrator
        e. SharePoint Administrator
    2. Business Applications Manager
        a. Developer, Grade 2
        b. Developer, Grade 2
        c. Database Developer, Grade 2
        d. Front-end Developer, Grade 2
        e. Front-end Developer, Grade 3
        f. Report Developer, Grade 2
        g. Report Developer, Grade 3
        h. Business Analyst, Grade 2
        i. Business Analyst, Grade 2

Maintenance Structure

a. IT Director
    ii. IT Desktop and Server Manager
        1. Database Administrator (DBA)
        2. Desktop Support Technician, Grade 1
        3. Desktop Support Technician, Grade 2
        4. CRM Administrator
        5. SharePoint Administrator
    iii. Business Applications Manager
        1. Developer, Grade 2
        2. Front-end Developer, Grade 3
        3. Report Developer, Grade 2
        4. Report Developer, Grade 3
        5. Business Analyst, Grade 2
        6. Business Analyst, Grade 2

Recommendation

Develop and comprehensive data governance policy. Increase IT staffing resources.

Conceptual Data Design

This section is intended to layout the conceptual framework and context of design recommendations made elsewhere in the document. It is not meant to be a formal design. ODOE should use approaches discussed below to direct detailed design processes.

Outline Data Model

Delaris recommends ODOE utilize a hierarchical data model like the one depicted below:

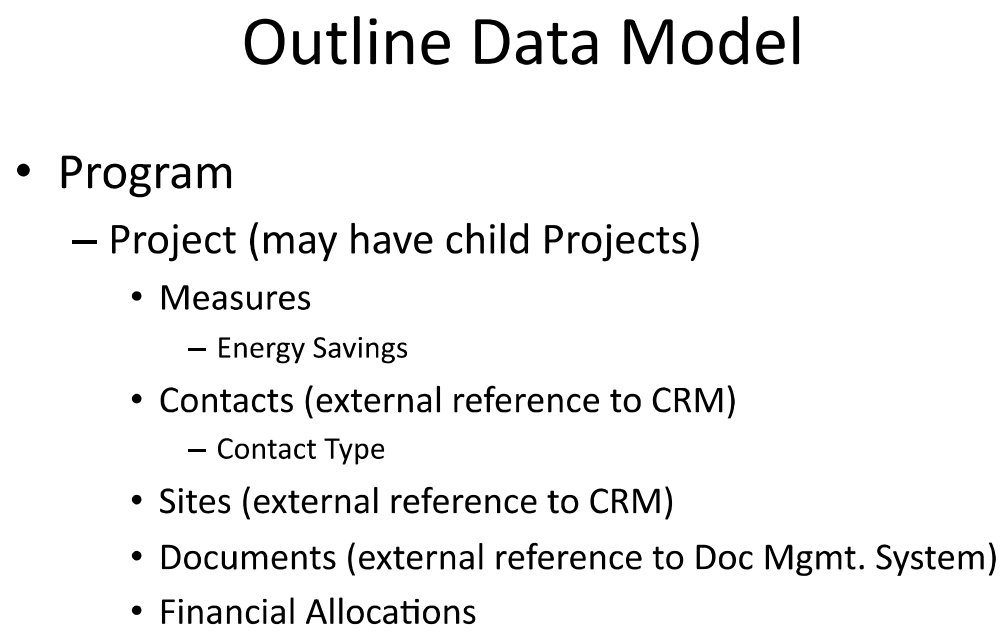

Figure 1: Outline Data Model Hierarchy of Objects

Interpreting the outline above, Program has-many Projects. Projects may have peer or child Projects through a self-referential many-to-many join that carries the relationship type. Projects has-many Measures. Measures has-many Energy Savings. Projects also has-many Contacts, which are accessed through a reference to an external CRM. Contacts carry a Contact Type. Project has-many Sites, which are also located in the CRM. Project has-many Documents, which are referenced through a DMS. Finally, Project has-many Financial Allocations, which may be of different types, including tax-credits.

This structure is built to allow summary data to flow upward through the data hierarchy for cross-program reporting and still provide tools necessary to construct separate program specific views of data. This model makes simple project reports by Contacts or Sites straightforward.

Systems Design

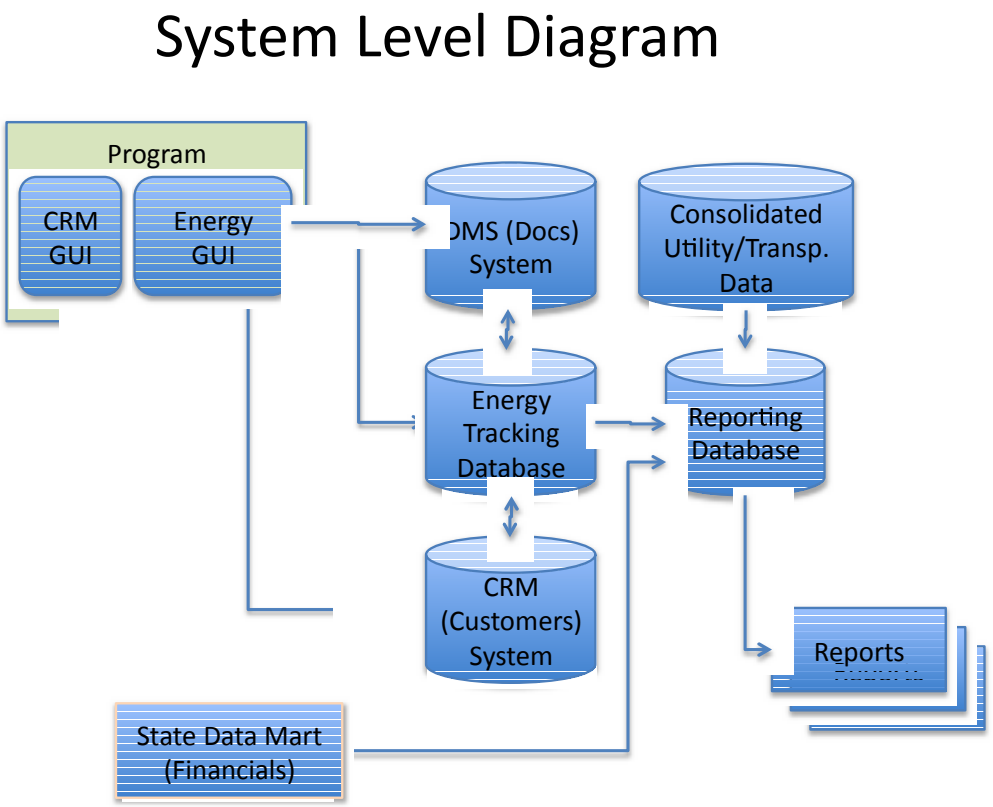

Figure 2: System Federation Diagram

This approach means "best-of-breed" systems can be used to manage different types of data associated with a project. The DMS stores documents, manages versions, and encapsulates document meta-data. The CRM stores Customer and Site data.. Financial systems (ODOE Data Mart feeds read-only financial data to the reporting database), and other ODOE stand-alone systems such a SELP and SEED have been excluded from this diagram. This reporting database separates the computational load of reporting from the OLTP load of the energy-tracking database. The reporting database will need to be periodically updated (once every 24 hours) from its upstream data sources. Also shown is a consolidated utility/transportation database, a future projection.

One implication of this data model is that effort needs to go into identifying and de-duplicating data. To make contact or site centric views accurate, duplicates of

contract or site records need to be merged. This linkage, across system boundaries, is more complicated than a classic CRM de-duplication scenario where duplicates are located in the database and the user chooses a winning record, causing data from the losing record to be copied to the winning record and deletes the losing record. If this happens in the data model described above, the losing record may have a reference inside a project in the energy-tracking system, which will lead to a dead-reference error when the project is accessed. To avoid this, de-duplication schemes must extend across systems and update the contact or site references on all projects so de-duplicated records are correctly referenced.

EAV Design

Entity Attribute Value (EAV) components are incorporated at the project and the measure level. EAV components must be incorporated at both project and measure levels because data is common between some projects within different programs, and different in others. Incorporating all variances of data into the actual data structures would cause both project and measure fields to get exceedingly wide, slowing the applications ability change. Because of this, Delaris recommends implementing EAV structures at both project and measure levels.

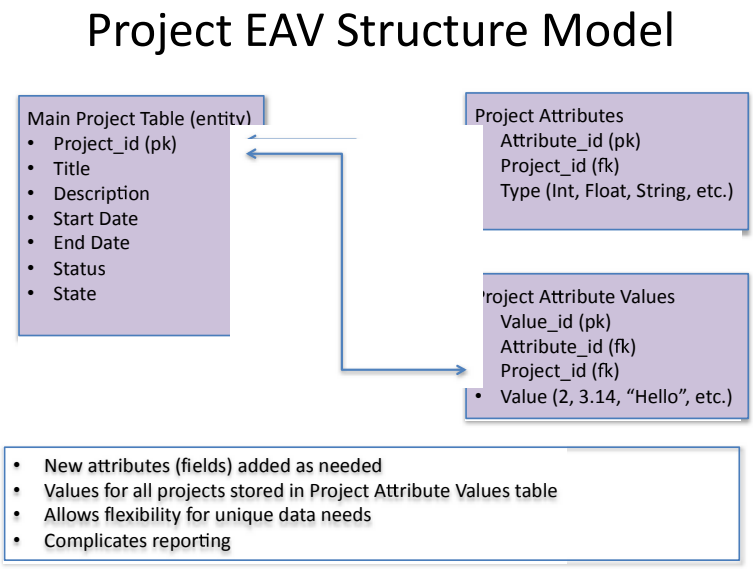

Figure 3: Project EAV Structure

This diagram shows how it could be incorporated at the measure level:

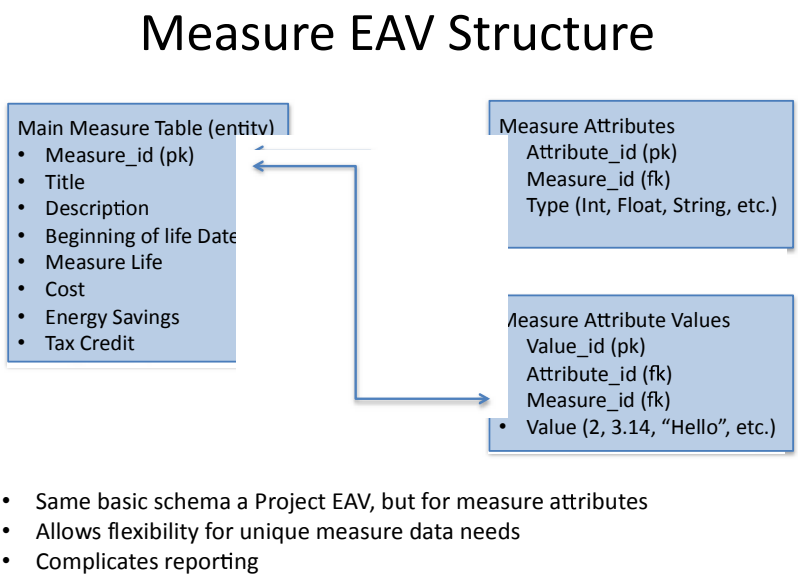

Figure 4: Measure EAV Structure

Each structure implements a sparse array structure. Instead of adding a field to the project table a project attribute is added, which carries the field type (integer, float, string, etc.). When a user enters a value for the field, a record is created in the project attribute values table, which carries the value of the variable. The same technique is applied to measures.

This design makes field addition quick. Fields are then carried in a template table (not shown) that are associated with a project type. When a project created, these project attribute template fields are copied into the project attributes table, pre-populating it.

The primary disadvantage with an EAV design is fields tend to proliferate, particularly without a strong data governance policy in place. Also, unlike conventional fields, EAV fields will not automatically show up in reporting tools without a specific query fragment to draw the variable out of the EAV structure.

Data View

An additional factor to consider is that users that are unaware of the differences between simple and integrated measure use in programs may be confused when they get unexpected results from ad-hoc queries. This typically is the result of a user trying to derive per-measure costs or savings from integrated projects, where the data does not exist on the measure level. This is an actual reflection of reality in industrial projects, but it can lead to some interesting and educational conversations about the differences between different types of programs and measures. We suggest hiding this level of detail from most users using SQL views. The Project level sums correctly in all cases, and this level of detail should satisfy most user ad-

hoc reporting requirements. Deeper dives into measure details and costs should be divided by well-designed views into a simple case and an integrated case to prevent confusion. Only qualified users should be given access to the deeper data structures, and peers that are aware of this subtlety in the data should review their work.

Recommendation

Implement and EAV design and form a strong data governance policy to limit field unnecessary field proliferation.

Simple vs. Integrated Measures

Energy management organizations like ODOE generally implement two types of programs, high-volume simple measures (e.g., RETC) and low-volume complex measures (e.g., EIP). Many "simple" measures are available in program like RETC, often projects have one or two, and measures have no interactive effects. In EIP, each project is custom and contains "integrated" measures; measures are installed as packages and have significant interactive effects with one another.

The simplest data model is to allow measures to carry savings data, as diagramed below:

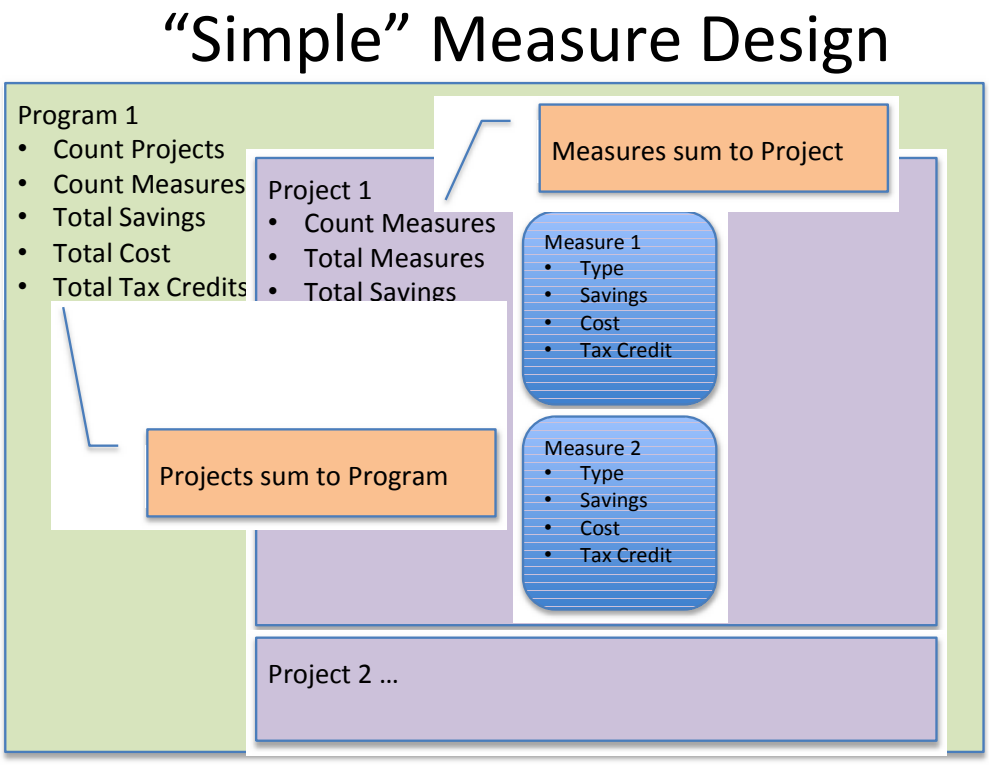

Figure 5: Simple measure design

Programs with "simple" measures designs are straightforward, savings, costs, tax-credits, measure counts can be summed by project and programs. Programs with "integrated" measures aren't as simple. Savings must be calculated on a per-project basis, as energy savings and cost can't be broken into measures, as the interactive effects measures have make measure-to-measure comparison across projects difficult. Instead, costs and savings are broken out into separate "savings only" measures, as diagrammed below:

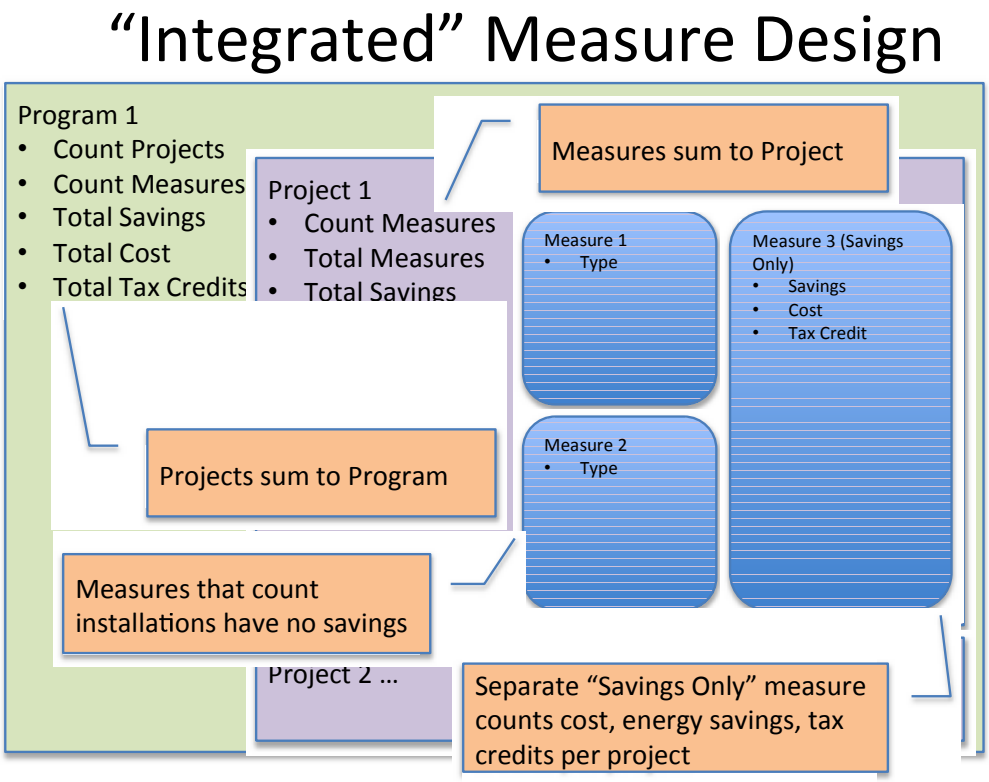

Figure 6: Integrated measure design

Individual measures are still applied to a project, but only carry measure attributes (type, count, "square feet of windows replaced"). Savings, cost, tax credit amounts and other per-project data are carried in a separate project-specific measure. This measure is not counted in the project sum to types and counts reporting, but is counted in the project sum costs reporting. The two classifications of measures allow the two types of data to be separated in the overall data structure and coexist without conflict. This measure delineation requires extra report writer training to make sure the right measures are included in reports. User with more limited permissions would only view a "masked" view of measures based on the level of reporting attempted. If viewing project level data all measure savings would be visible but when viewing measure level reports, savings from integrated measure packages wouldn't be shown.

Recommendation

Implement both "simple" and "integrated" measure designs. Provide report writers additional training on measure inclusion.

Implementation

The following is a description of our recommended phasing of a central data system implementation project.

Note: Phase 1 and 2 are interchangeable.

Phase 0: Present State

ODOE employs a number of different software solutions to track customer, application, and energy data. Most of these solutions are program specific stand-alone systems.

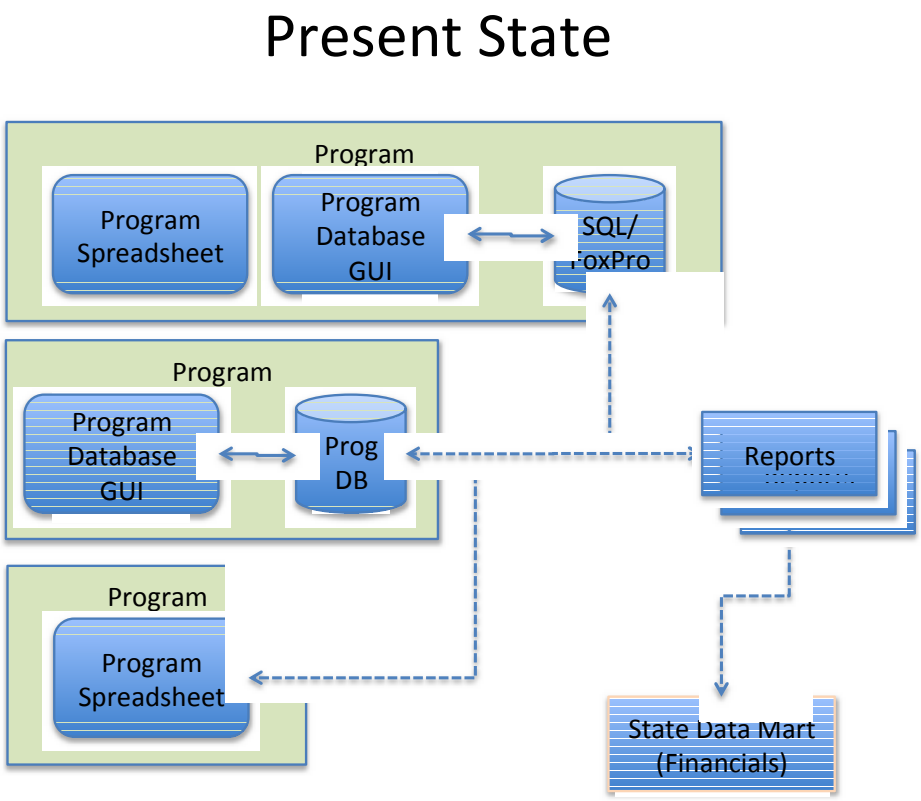

Figure 7: Present State

Systems/programs are loosely summarized through multiple manual reporting processes. Definition of data between programs is inconsistent; some projects are double-counted between programs. Standalone program data isn't accurate without adjustments.

Post Phase State

Current state, no changes have been made.

End-state risks

Risks Mitigated

None

Risks Remaining:

All

- Low data quality
- No data governance
- Process isolation
- Decentralized data collection
- Obsolete technology
- Evaluation Support Missing

Phases Overview

- Phase 1 – Report Server
  - Programs still have little commonality, but reporting server enforces common data definition. Data quality still enforced after entry. Some tactical reporting. Annual reporting via canned reports from Reporting Server.
- Phase 2 – Decentralized Data Entry
  - Programs use common systems, enforcing common data definitions and processing. Data quality enforced at data entry. Improved tactical reporting, addition of operational
- Phase 3 – Integrate Other Programs
  - Remaining programs integrated into new system.
- Phase 4 – Integrate Utility/Transportation Data
  - Form partnerships with serviced utilities, obtain utility data on a monthly download basis. Data consolidated and anonymized, made available for internal and external energy savings research

Phase 1: Implement Report Server

Create reporting server and develop prioritized set of reports. Focus report development to performance of ODOE, capturing historical data, and contributing to the Annual Energy Savings report. Integrate program data into the reporting database through automated machine-readable tabs in established program tracking spreadsheets or through direct integration Extract Translate Load (ETL) scripting between existing program databases and the reporting database.

Implement a reporting database with a data-warehouse (star and spoke) schema. Do not optimize the data warehouse for On-Line Transaction Processing (OLTP).

OLTP database designs put a premium on responsive user interactions at the cost of clarity within the database schema.

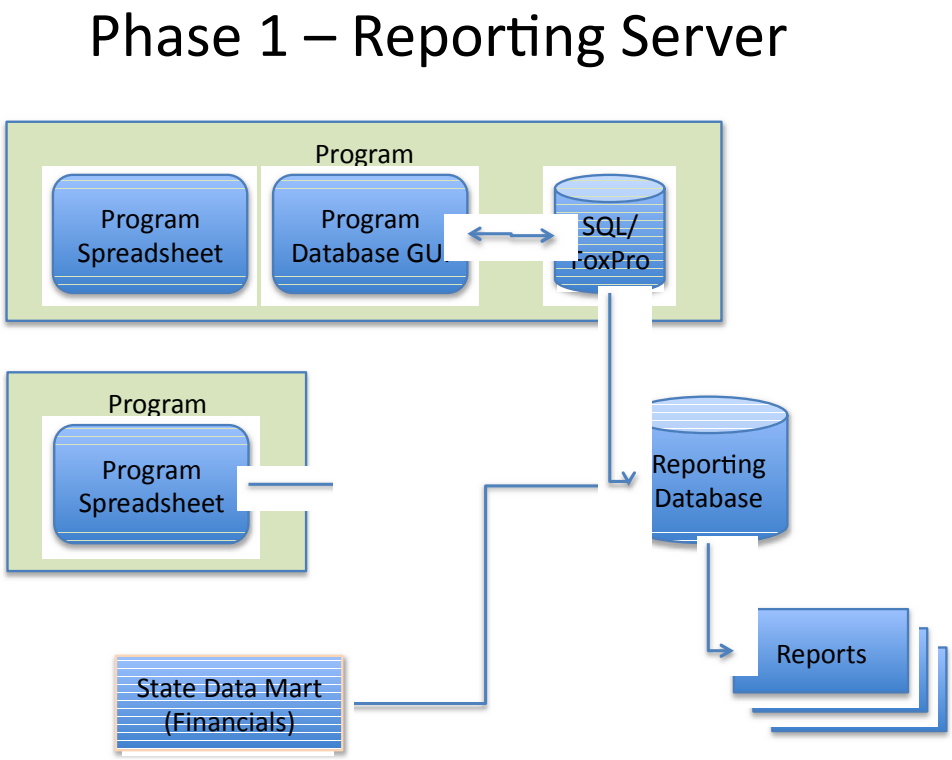

Figure 8: Phase 1 - Reporting Server

Post Phase State

Reporting SQL server in place, some high-level reports developed. Programs still have little commonality at data-entry, but output to a common data structure, beginning the transition to overall data governance and common data definitions. Annual Energy Savings report is systematized.

Note: this phase is interchangeable with Phase 2.

End-state risk

All risks are still present, but some are reduced.

Risks Mitigated

- Beginnings of data governance, risk reduced
- Beginning to improve process isolation, risk reduced

Risks Remaining

- Low data quality
- Some data governance risks remain

- Process isolation less, but risks remain
- Decentralized data collection
- Obsolete technology
- Evaluation Support Missing

Phase 2: Centralize Data Entry

Implement two sub-systems concurrently: a CRM system, and an energy-tracking package. Integrate the CRM package with the energy-tracking package. Implementation should follow the general procedure below:

1. **Software Test and Validation:** Configure the software suite to incorporate specific program needs. This includes developing a small set of data quality reports that can be used throughout the process to monitor data integrity as data is loaded into the system and the system is brought online.
2. **Data upload:** Existing data is uploaded into the Energy Tracking Package and data in the OLTP and reporting subsystems are checked.
3. **Pilot use:** Particularly with early usage, selected users test the system using a representative set of test cases. If desired, a combination of old and new data collection systems are used in parallel. Data is entered into both, for a period of time, and the results are checked with one another using tactical data quality reporting. If necessary, final corrections are made and tested.
4. **Program cutover:** When program management has developed enough confidence in the software, all aspects of the program are cut over onto the new system.
5. **Evaluation and maintenance:** After a period of operation, the software and cutover process are evaluated and the lessons learned incorporated into the next release of the software and the next program cutover.

Phase 2 – Centralized Data Entry

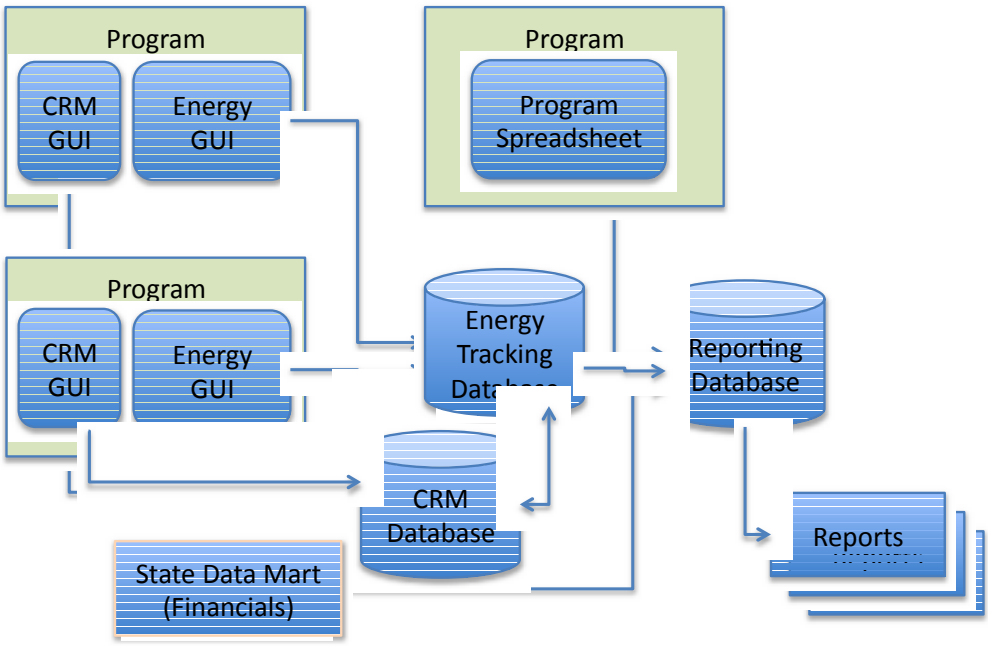

Figure 9: Phase 2 - Centralized Data Entry

Post Phase State

Some, programs use common customer and program tracking systems. Data quality is enforced at entry (for those with program and customer tracking systems) and common data definitions across programs are enforced. Tactical reports are developed and implemented.

End-state risks

All risks described in the risks portion of the document are still present, less specialization and fewer single-points-of-failure among personnel.

Risks Mitigated

- Data governance among programs using common software, risk greatly reduced
- Greatly decreased risk of process isolation using common software, risk greatly reduced.
- Much higher data quality for programs using common software

Risks Remaining

(The following risks remain for programs not on common software platforms)

- Low data quality
- Data governance risks remain

- Process isolation
- Decentralized data collection
- Obsolete technology
- Evaluation Support Missing

Phase 3: Integrate Remaining Programs

Complete integration of remaining ODOE programs with central data systems.

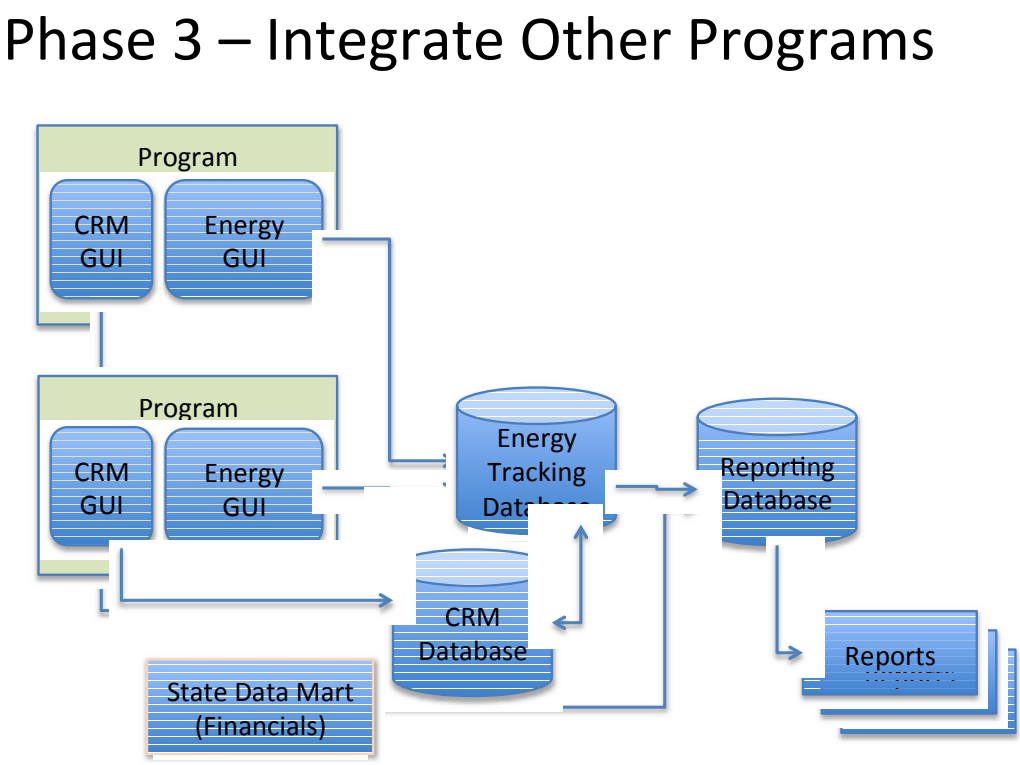

Figure 10: Phase 3 - Integrate remaining programs

Post Phase State

All programs use common customer and program tracking systems. Data quality is enforced for all programs at entry and full cross program data definitions are enforced. Tactical reports are used throughout ODOE to monitor data quality. Operational and strategic report development is ongoing.

End-state risks

Risks Mitigated

- Data governance
- Process isolation.
- Data quality
- Decentralized data collection

- Obsolete technology

Risks Remaining

- Evaluation Support Missing

Phase 4: Integrate Utility and Transportation Data

Establish agreements with served utilities and sister energy organizations to gather and consolidate energy data. Use centralized energy data to better understand strategic decisions, evaluate program effectiveness, and track metrics toward statewide and agency energy goals.

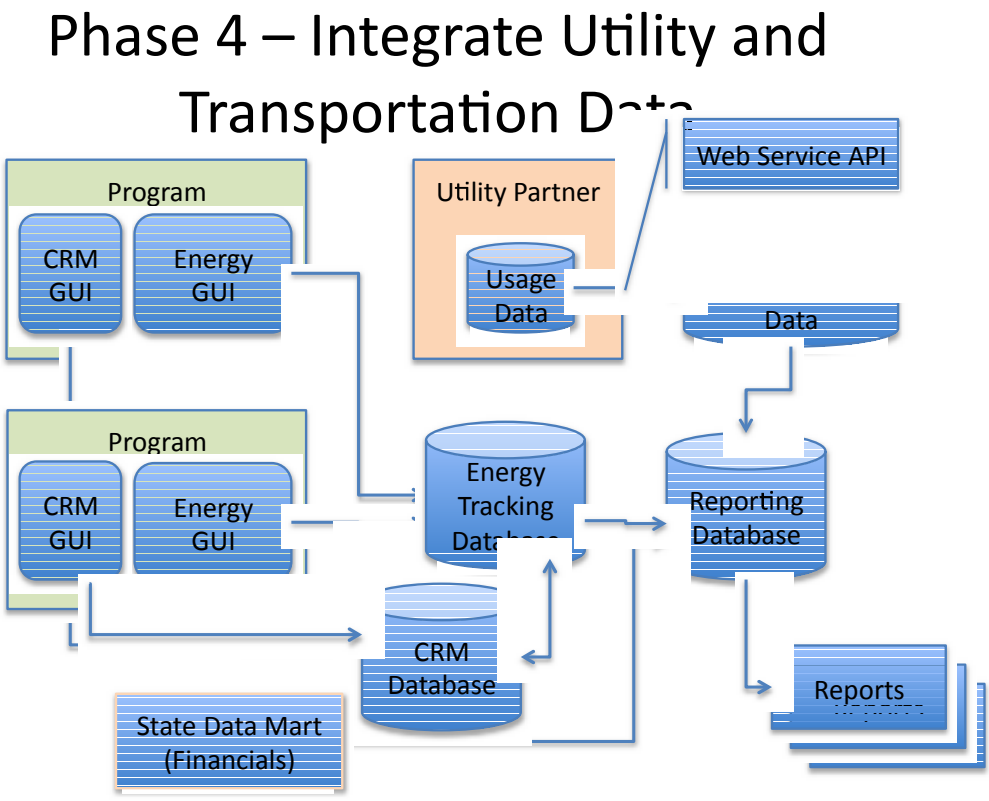

Figure 11: Integrate Utility and Transportation Data

Post Phase State

Agreements are in place with all external energy entities to collect and consolidate data into a single Utility/Transportation data warehouse. External utility and transportation system data is incorporated into a central data view. Summarized and anonymous data made available to external entities (academic energy researches, commercial energy partners, program evaluators) under controlled agreements for research.

End-state risks

Risks Mitigated

- Data governance
- Process isolation.
- Data quality
- Decentralized data collection
- Obsolete technology
- Evaluation Support

Risks Remaining

- None

Architecture choices

There are a number of architectural choices ODOE will face as it moves forward with each phase described above. This section is not intended to be an exhaustive review of all available options, but rather as a list of explored options. Because most of ODOE's existing infrastructure is likely to be phased out, this project provides a unique opportunity to revisit basic assumptions made in previous projects.

Commercial Software Packages to Manage Energy Incentive Programs

In recent years, a number of efforts have been made to fit the Energy Savings/Measure Management business processes into a standard set of business software packages. The Energy Trust of Oregon tried Epicor[14] and PECI SalesForce. Both efforts have been unsuccessful to date.

Sources of Failure

Delaris believes multiple factors led both projects failure:

- *Measure Simplification* - There is consistent temptation to oversimplify what a "measure" is during the requirements gathering and system selection phase. While the energy incentive business process is unique, traits are often similar enough to a business purchasing processes that it is easy to confuse a measure with a "product" or a "purchase order." This is particularly easy during pre-sales, when the customer may not have full awareness of the problem nuances and sales-engineers over-sell the capabilities of their software. The full complexity of the problem and consequent costly modifications to the base system that are required only becomes apparent after the sale is made and development is well underway.

- *Existing System Architecture* - Existing system frameworks impedes solution implementation. Modern Enterprise systems place a premium on

[14] http://www.epicor.com/pages/default.aspx

configurability rather than customization; meaning developers are required to modify the system to fit unique business processes.

Configurable software generally retains a core workflow that is incorporated into its base system source code. Venturing outside of these boundaries, modification and maintenance become *very* expensive. Delaris has observed specific instances of such systems being modified to handle energy incentive measures; requirements are just far enough outside what enterprise software packages are designed for that the additional cost and complexity required for correct measure function becomes cost prohibitive.

- *Fragility of Customized Systems* - Implementations of heavily customized systems tend to be very fragile. Complex dependencies between core enterprise code, third-party modules, customized code, and configurations, often crossing several application domains (core, module, layout templates, database configuration, database structure, etc.), demands high-level internal expertise or consultants to maintain. Routine updates to core or third-party sub-module code can cause errors as well.

  While enterprise level software is designed for general purposes, Delaris' experience shows incorporating energy measure management consistently moves these systems beyond their reasonable capabilities.

Recommendation

ODOE purchase or build a purpose-built energy management system. If ODOE elects to incorporate energy management into another set of enterprise software (i.e., CRM or ERP), then extensive requirements/system function analysis should completed before start of implementation.

Closed vs. Open Source Tools

Toolsets

As ODOE works within the State of Oregon IT environment, commercial enterprise software (i.e., Oracle, Microsoft) tends to be favored. While these are indeed robust technology platforms, Delaris believes there is opportunity within open source tools simply unavailable with competing closed source software packages.

Commercial Products

Primary arguments for commercial software products often center on the definite support and liability chain. If something goes wrong with a piece of software, and a support contract is in place, then the manufacturer is required to fix it. This support

was traditionally unavailable within the open source world. Recently, however, open source quality has started to exceed that of their closed source counterparts.[15]

Delaris feels long term pressures from open source solutions will disrupt the closed source software market considerably. Delaris feels this may cause long-term supportability issues for applications based on closed source platforms.

Recommendation

Objectively study current trends in open and closed source enterprise software. MS-SQL is a great product, but there are equivalent open source products that may be less expensive and may provide a better long-term support path.

Full Relational vs. EAV Data Models

EAV data model popularity has increased in recent years with demand to make software more configurable than customizable. Classically programmed applications require a developer to "plumb" a new field definition through multiple layers of code (place it on the screen, add it to the database, etc.). In an EAV configurable application (e.g., SharePoint), users edit a screen and the field is automatically added to both the screen and the data model without changing the layout of the database.

> *Configuration* - Allowing a user or administrator to change the behavior of a program from within the program itself.
>
> *Customizable* - Utilizing specialized expertise (i.e., a developer) to modify source code to alter a programs behavior.[16]

EAV Advantages

EAV applications are very flexible, power users and administrators can build and change applications rapidly.

EAV Disadvantages

EAV system flexibility comes with performance and complexity penalties, commonly manifested during reporting. EAV systems also rely on a data extraction process to "flatten" data within a reporting server.

EAV systems also require some level of application level "accounting," for variables and types, that leads to system complexity. In a classic system, a field named "kwh_per_year" might be defined directly in the database as a floating-point number in a "measure" table. In an EAV system, the variable "kwh-per-year" may instead be

[15] http://www.coverity.com/press-releases/coverity-scan-report-finds-open-source-software-quality-outpaces-proprietary-code-for-the-first-time/
[16] http://en.wikipedia.org/wiki/Entity–attribute–value_model

stored as a string in a general "variables" table, and converted to a floating-point number when referenced.

The configurability EAV systems provide can introduce unnecessary complexity as users modify them in an ad-hoc manner. Such modifications can conflict; for this reason, organizations using EAV type systems must exercise some degree of control over additions and modifications to the system made by users.

Recommendation

Implement a relational/EAV hybrid data model. Utilize traditional relational database design to store core data fields (e.g., general project data, locations, etc.) and implement an EAV structure to manage program specific project and measure attributes. Implement a configuration control process to assure data configuration changes to EAV subsystems are reflected correctly in reports.

Relational vs. Non-Relational Databases

Relational Databases

Relational databases organize data in tables that "relate" to one another. [17] Efficiencies with these databases are experienced when updating a field relating to a large number of records in another table. In addition, because only a single instance of a data field is stored, disk space is conserved. This architecture is met with disadvantages.

Relational databases segment data across many tables, which result in computationally expensive reporting operations. Particularly for "tall" and "narrow" tables (i.e., tables with few columns and many rows or tables with many columns and few rows), the efficiencies of relational models during read and write operations degenerate as tables grow very large. Energy Trust has utility data tables with over a billion records.

Non-Relational Databases

Non-relational databases or "No-SQL" databases like Mongo-DB, Cassandra and HBase,[18] were developed to solve problems associated with storing massive amounts of data. Data is stored as flat objects reducing database computation requirements.

Recommendation

That ODOE investigate a No-SQL solution for consolidation and storage of larger data sets.

---

[17] http://en.wikipedia.org/wiki/Relational_database

[18] https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models

Contracting Strategies

Procuring large, complex software is a challenge involving change in almost every aspect of the organization. To Delaris' knowledge, **ODOE has never completed a similar project.** Delaris believes ODOE does not possess the historic institutional knowledge, internal technical skill, or internal management depth to manage a project of this scale successfully. Delaris recommends specific procurement and contracting strategies to alleviate some associated risks.

Multiple vs. Single Contract Strategy

Single contractor selection means having one organization responsible for the entire system. This approach avoids inter-contractor conflicts, often difficult for a customer to manage. In addition, this procurement strategy can obscure view of internal product development well after test and delivery.

Multiple contractor procurement generally exposes internal product weakness earlier. Contractors work off each other's constructive criticisms, making the whole system stronger in the end. However, significant management overhead is required to make this work.

One software development methodology that has developed to mitigate problems associated with single contractor procurement pull the customer/contractor relationship closer together forming an Integrated Product Team (IPT).[19][20] IPT's seek to combine team members into a single governing body, providing increased project transparency assuring team members with appropriate skill and perspective are always involved in the decision making process.

Agile software development is another popular approach.[21] Agile is sometimes articulated by 12 guiding principles:

1. Customer satisfaction by rapid delivery of useful software
2. Welcome changing requirements, even late in development
3. Working software is delivered frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the principal measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Self-organizing teams

[19] http://en.wikipedia.org/wiki/Integrated_product_team
[20] http://www.acq.osd.mil/se/docs/DoD-IPPD-Handbook-Aug98.pdf
[21] http://en.wikipedia.org/wiki/Agile_software_development

12. Regular adaptation to changing circumstance

The central concept is customers should be involved throughout the development process, incorporating rapid change and feedback into the evolving product.

Delaris believes ODOE doesn't current have management resources to embark on a multiple contractor strategy without hiring or contracting additional resource.

Recommendation

ODOE utilize a single contractor procurement strategy, mitigate risks associated with transparency by instituting an IPT model with Agile development methodology throughout.

Fixed-Price vs. Cost-Plus Contracting

Fixed-Price

With fixed-price system procurement, requirements are clear, and the end goals are well defined. Clear statement of the problem is written alongside clear descriptions of the surrounding environment, interfaces, growth potential, training, timing, and sizing. Contractors propose solutions and a proposal is accepted. Acceptance criteria are negotiated up-front and written up front. Large portion of the payment is withheld until the system passes acceptance testing. Following acceptance tests, discovered bugs may or may not be fixed based on the warranty agreement.

From a customer perspective, fixed-price has the advantage of defining a set product for a set amount of money in a specific amount of time. Advantageous from a budgeting and scheduling perspective, if needs change or new needs are discovered during development, change order processes must occur which can be onerous.

From a contractor perspective, fixed price is about managing risk. Provided system scope and deliverables don't change, the contractor is in good shape. Often times requirement interpretation can change, particularly with projects with long time lines. Disagreement over project scope can lead to a combative contractor/customer relationship.

The second issue for the contractor is miss-estimation of the problem and/or the occurrence of unknown development issues. In both cases, the contractor is liable for cost overruns. To prevent this, contractors typically build in a "pad" to a fixed estimate in order to account for unknowns. Raises the cost of the procurement, sometimes significantly.

Cost-Plus

In a cost-plus model, the customer has a loosely defined problem and/or loosely defined goals. RFPs contain less detail, and contractors bid a rate and a loose time schedule. Generally, the first part of the project is a more detailed exploration of goals and requirements, followed by one or more phases of development and delivery. Final testing phase definitions are more static.

From a customer perspective, cost-plus procurement allows the project to take advantages of "targets of opportunity" as they occur. As changes are discovered, they are incorporated into the schedule without the formality of a change-order process. Customer/contractor relations are more cooperative because of this.

The problem with cost-plus procurement from a customer perspective is that the customer wears the risk of changes and unknowns. The customer is essentially self-insuring. When changes occur, it is up to the customer to manage them, both from a budgetary and a schedule perspective. This can lead projects into a "wandering in the woods" phase, where no one is sure when the project is done.

From a contractor point of view, cost-plus has many advantages. First and foremost, the contractor is not wearing the change risk. This allows the contractor to bid closer to what they think the actual cost of the project will be rather than tagging on a pad for unknowns. This is particularly advantageous where new software or technologies are being developed. On the downside, the contractor profit is fixed up front, there is no incentive to get the job done faster or cheaper in order to maximize the profit under fixed-price.

When procuring software, unless the software is a packaged system that has been deployed and tested before, the procurement is more like a technology development contract than a technology purchase contract. Every situation is different and which contracting form to use comes down to three factors:

1. Is the problem well defined? (Does software already exist to solve the problem and can you buy it?)
2. Does the organization have clear end goals in mind?
3. Does the organization have the internal depth and skill to manage a formal change-order process?

If the answer is "yes" to all three questions, then ODOE should choose a fixed-price contracting strategy. If not, then ODOE should consider using cost-plus vehicles for portions of procurement with the most unknowns while setting clear limits in cost and schedule to prevent the project from going astray.

Recommendation

ODOE spends additional time to define the problem in more detail, develop clear project goals, and develop the internal management processes and expertise to write and manage a fixed-price software procurement contract.

Multiple Small Projects vs. Single Big Project

Statistically, smaller projects succeed more than larger software projects. There are a few reasons for this, but the most pertinent have to do with complexity and communication.

Larger projects tend to be more complex and result in radical change. Development staff is organized into teams, each developing "sub-systems" to address a subset of the problem. Management is responsible for making sure sub-systems are compatible with one another.

Larger projects also have longer time horizons, months and years as opposed to weeks and months. Truth changes over time. Personnel move on, technologies advance, and organizations change, sometimes causing requirement to become obsolete. Each of these factors requires management.

This leads to the second big problem, which is communication. As stated before, people in big projects tend to work on isolated teams. Managers become the information conduit between teams. Because of the interrelated nature of complex systems, a problem or a change in specification can have exaggerated effects elsewhere in the system. It takes a certain degree of management skill to identify potential problems, project their impacts, and resolve the issue.

Smaller projects, tend to consist of one team that meets regularly. Interfaces between sub-systems are few and more easily managed. Knock-on effects of errors are easier to identify and fix. Advantageous small projects provide often make it more efficient to break larger problems down into several smaller projects. This approach allows delivery of multiple progressive tangible results as progress is made.

The disadvantages of breaking a large project into smaller projects come from time and money constraints. Small-project strategies take longer as each contain their own procurement cycle, teams, and sometimes contractors.

The other disadvantage is the progression of small projects, may cause the main project to run out of money leavings incomplete project goals. There is not the impetus to drive to full completion that exists in a larger project.

Recommendation

Delaris recommends pursuing a single large project strategy to implement a central data tracking system.

Cost

Cost will vary greatly depending on strategies ODOE chooses to pursue, and the availability of software to buy, license, or modify. Delaris has attempted to research the current energy management landscape in order to recommend a path forward. What follows is first a general discussion of the basic options, followed by our recommendations on how to proceed.

Buy vs. Buy and Modify vs. Contract Build vs. Internal Build

Buy (probably not possible)

The most desirable solution is to buy the software off-the-shelf if possible. There are a number of packages to do energy. Extensive market research was not included in Delaris' scope and thus only a cursory look at each systems capabilities is provided. We believe that ODOE should undertake more extensive market research before deciding on a final solution.

Buy and Modify (best odds of success)

Delaris believes this is ODOE's best option however wherever ODOE chooses to purchase a product from, clear ownership of various pieces would need to be established and licensing arranged before this solution could be used.

Contract Build (diminishing odds of success)

If there are no commercial packages suitable for purchase and if clear ownership cannot be established for pre-developed solutions, then Delaris recommends ODOE contract with a database application development contractor to build the recommended solution. This option will be the most expensive. Delaris recommends ODOE proceed with caution with large software development projects. They are notoriously difficult to estimate and manage. In addition, custom developed software requires a long (years or decades) commitment to maintenance. We recommend a series of small projects to diminish this risk, as well as using Integrated Product Teams and Agile development to keep the project on track and involve the whole organization throughout the development cycle.

Internal Build (least desirable)

Because of budget and time constraints, ODOE may decide to attempt to build some or all of the software using internal maintenance programming resources. Delaris recommends against this approach. ODOE has attempted this in the past with limited success. Current IT staff are primarily tasked with keeping ongoing programs up and running, distracting them from what turns into a secondary task of developing new software. Maintenance programmers tend to use existing, familiar technology even if it is obsolete, thinking that they are more efficient in familiar tools even though the tools may be at a technological dead end. Finally, good software is typically produced by small multidisciplinary teams. Most organization do not have the manning overhead to constitute such a team, nor the willingness to train and maintain them. Additionally, ODOE's experience in managing software specification, development, and testing is limited, and should be hired or developed

internally to be successful. Finally, the pool of potential developers and managers in Salem, Oregon is thin, with truly many qualified developers and development managers already fully engaged. For all these reasons, Delaris believes this option has a minimal chance of success.

Budget estimates

The following are some budget estimates for the various procurement options outlined above. Where appropriate, Delaris provided a level of certainty.

Buy - $300K to $500K

Main cost drivers in the "Buy" option are licensing costs and implementation costs. Delaris has doubts as to whether one of the current crop of off-the-shelf solutions is truly going to meet ODOEs needs without alteration, which would drive ODOE towards a "Buy and Modify" solution.

Buy and Modify - $1.3 million to $1.5 million

There are a number of unknowns in this strategy that make it difficult to estimate. Generally, it's reasonable to assume that this option is going to fall somewhere between a pure buy and a pure build. The driving factors are 1). Which package is ODOE going to buy? 2). How much configuration and customization is needed to make the package fit ODOE's needs? If ODOE selects packages available from specialty vendors licensing fees to can be expected to fall between $300,000 and $500,000. Delaris estimates required modifications (program implementation, data upload, modifying) will take 3-member team 18 months to modify the product. Costing additional $1 million.

Contract Build - $1.5 to $2.5 million

This solution is easier to estimate based off previous experience. Previous projects Delaris has been involved with required two years and engaged a development staff of five, including a manager (10 man/years). Working with a loaded hourly rate of approximately $125 per hour x 8 chargeable hours per day, leaving the project burn rate at $1,000 per developer per day.[22] Assuming a team size of 5 developers x $1,000 per day = $5,000 per day. Using a working figure of 250 working days per year 250 days x $5,000 per day = **$1.250 million per year**. Two years comes to approximately $2.5 million. Delaris's experience has shown that efficiencies identified during the estimation processes are often consumed by scope creep and project unknowns. For this reason, Delaris believes that $2.5 million is an accurate estimate.

Internal Build - $1.5 million

Using a similar methodology, Delaris can estimate the cost of using internal development resources. Given a loaded rate of approximately $150K per year per developer and assuming two developers working at half efficiency due to existing workload, and assuming they are going to cover the same ground as the five-man

[22] http://www.leadingagile.com/2012/11/calculate-budgets-agile-team/

development team above, this means they generate one loaded man-year of work costing approximately $150K per year. Based off this projection, the project takes approximately 10 years to complete and costs about $1.5 million. ODOE can choose to shorten the time horizon by hiring more developers or can choose to scope the requirement down to size the problem to two developers in two years (a two man-year problem), but both of these solutions present additional development problems. This analysis does not consider the loss of efficiency and distraction caused by developers jumping back and forth between projects in an attempt to keep current systems running while at the same time trying to develop a new system. ODOE has already tried this technique with limited success.

Possible Synergies With Other Organizations

Other organizations within Oregon do similar work. It might be possible to cooperate with one or more of them to reduce costs.

Energy Trust of Oregon (ETO)

The ETO energy-tracking system closely fits ODOE's requirements. Developed over the past 14 years, ETO continues to devote time and resources to improving their solution throughout its life. Delaris recommends ODOE keep lines of communication open with ETO as there are a number of business-to-business data sharing opportunities that should be encouraged. Data processing systems capable of sharing data back and forth across a compatible set of interfaces has a lot of potential to improve the energy understanding for the State of Oregon.

Northwest Energy Efficiency Alliance (NEEA)

NEEA encourages changes in the marketplace via information, mindshare, and attitudinal factors. Their programs are necessarily softer (no financial transactions for savings) and longer term. This changes their data tracking needs significantly. However, they do have a spreadsheet reporting model that could be used as a prototype for a short-term solution for ODOE's reporting needs. Delaris recommends ODOE evaluate NEEA's solution as part of the larger recommended solution, perhaps as an intermediate step or adjunct capability for small programs. The NEEA package may be purchasable, and the costs of implementation should be minimal.

Methodology

The Oregon Department of Energy RFP for this project expressed Agency desire "to gather, aggregate and analyze energy data to become the central source of energy information in Oregon. Creating of the Oregon Energy Database…to support the Agency's mission is to promote a safe, clean, sustainable energy future for Oregon".

The RFP explained the project purpose: "to identify the business requirements, potential implementation methods—including business processes—and other essential information critical to the creation of the Oregon Energy Database". RFP tasks included, but were not limited to:

- Identifying energy data sources
- Identifying potential database users
- Identifying analytical needs
- Identifying reporting needs
- Identifying technical alternatives to meet Agency's goal of creating a central source of energy information in Oregon, and
- Providing recommendations and estimated costs for paths to implementation

Phase Descriptions

Kickoff

Delaris met with project stakeholders to review the proposed project plan and validate the organization of project work groups and a proposed schedule of discovery meetings. An ODOE-wide kickoff meeting with all workgroups and project stakeholders, was held, to provide information about the project, the Delaris approach, and how they will be asked to participate.

Discovery

During the Discovery phase of the project, Delaris managed a five-step process with each workgroup, exploring the unique interactions, processes, services, and data needs of the ODOE programs and their associated management processes.

Pre-Interview Survey

Prior to the in-person meetings, Delaris created a web-based survey to facilitate constructive thinking, uncover meaningful data examples and deepen staff understanding of the Oregon Energy Database project.

Interview

Interviews and working sessions with ODOE staff were scheduled by work group and organized by program and critical process. During these sessions, Delaris began collecting artifacts, policy documentation, and top-level business requirements. Meetings were held with:
Office of the Director

- Human Resources/Communications

Energy Development Services
- Business Energy Incentives
- Energy Development Services
- Energy Tax Credits, Rebates, and Compliance
- Energy Loan Program
- Energy Facility Siting

Planning and Innovation
- Energy Conservation
- Energy Technology

Central Services
- Budget/Finance & Operations/Information Services & Technology

Additional interviews were held with a group of external data sources and consumers, including the Energy Trust of Oregon, the Northwest Energy Efficiency Alliance and the Northwest Power and Conservation Council.

Data analysis and modeling followed each interview/workgroup session, during which Delaris developed visual representations—"models"—of existing data, who uses it, and how the program and process managers refer to it. The data models are a compilation of data dictionaries, data flow processes and high-level entity-relationship models and focus upon the sources, translations, and destinations of data most relevant to ODOE staff and stakeholder needs.

Deliverables:

- Identified internal and external data sources.
- Identification of redundant data sources within ODOE
- Identification of database users

Validation and Metadata

After outlining the various data models discussed with each group, follow –up meetings were held to validate initial findings, to make additions and refinements and to expand the collective understanding of each programs and work group's data.

During a third meeting with each workgroup, Delaris helped the participants uncover facts about how their data is stored, how it ages, how it might be indexed for effective search, and how users plan to summarize the data in reports (metadata). As in the initial discovery phase, these meetings were followed by another session of data analysis, enabling Delaris to create new data models and to refine existing models.

Deliverables:

- Identified analytical needs
- Identified reporting needs

Validation and Risk

A fourth meeting was held with each group to discover the business criticality, exceptions, security requirements and licensing of data. Following this meeting, Delaris made final edits to each groups' and program's data models.

Deliverables:

- Classified data security requirements

Document

The data models developed during Discovery were analyzed and combined across groups, creating an organizational data model, outlining how the Agency intends to utilize the Oregon Energy Database.

Delaris outlined strategies to improve efficiency (de-duplicate data) and security and began identifying tools and technologies that will help the Agency streamline operations.

Agency-wide data model and functional/non-functional requirements were recorded in a database requirements document, which was reviewed (over three iterations) with Agency technical and non-technical stakeholders, with Delaris editing as directed.

Recommend

Delaris identified multiple implementation paths and provided a comparison of each solution's business, technical, and financial limitations in the final report.

Deliverables

- Identified risks and barriers to implementation
- Identified technical alternatives
- Prepared documents that may be required to implement the Oregon Energy Database
- Provided recommendations and estimated costs for potential paths