



Whitemarsh
Information Systems Corporation

Data is Executed Policy

A Case Study

Whitemarsh Information Systems Corporation
2008 Althea Lane
Bowie, Maryland 20716
Tele: 301-249-1142
Email: mmgorman@wiscorp.com
Web: www.wiscorp.com

Table of Contents

1	Rationale for “Data is executed policy”	1
2	Case Study	4
2.1	The Case Study’s Enterprise	4
2.2	The First Attempt	5
2.3	The Assessment	6
2.3	The Plan	8
2.4	The Prototype Phase Results	8
2.4	The Implementation Phase	9
3.	Lessons Learned	10



1 Rationale for “Data is executed policy”

Quality database is an expression of organization, clarity, and precision. It may or may not be computerized. If it is, it may exist on a desktop, a server, or a large mainframe. Finally, a database may or may not be centralized. Notwithstanding the mode, the mechanism, or the form of database implementation and operation, the codification of and adherence to data semantics, which are the rules for meaning, validity, and usage, are prerequisite to a successful database.

When a database is on a computer, it represents the automation of the knowledge component of a business, which is manifest through the business's quality operation, planning and management. With quality database, management of a business can research the past, organize the present, and plan for the future. To have quality database is to have defined policy because,

- ! Each database object class's data structure within the database is the data representation of a policy's definition.
- ! The database object class's processes through which rows are added, deleted, and modified are the mechanisms necessary for policy execution, that is, the policies procedures through which database objects are transformed from one valid state to another.
- ! A fully defined policy includes both its complete definition and its necessary steps for coherent execution.
- ! Interrelated collections of rows across multiple tables of one object class and across multiple database object classes form more comprehensive policies.

When data is seen as executed policy, and is realized through database object classes, then quality databases support following within the enterprise:

- ! Business information systems that are a coherent union of the policy and then execution of the procedures that represent the accomplishment of the policy.
- ! Consistent collection and/or modification of policy instances through the life cycle of the policy.



Data is Executed Policy, A Case Study

- ! Consistent execution of policies when ever, where ever, and how ever deployed as the essence of the policy and the totality of its critical procedures are encapsulated within the database object class itself
- ! Minimized redundancy and consistent policy implementations across distributed environments as the database object class can be distributed through encapsulated strategies
- ! Comparable instances of deployed policies as they are independent of hardware architectures and operating systems

But, what forms the basis of a database object? Simply, it is a business' policies and procedures. While policies can exist without procedures, the converse is not true. This ontological priority dictates that procedure is dependent on policy. Not only, in this case, do they go together like hand and glove, the glove (procedure) serves no useful purpose without the hand (policy).

A database object is a person, place, or thing that has internal consistency, and is transformed from one valid, predefined state to another through well defined rules. The minimum value states are null and valued. The internal behavior of a database object class as it transforms database objects from one state to another is immaterial to its user. Database object classes conform to the requirements of business rather than the converse.¹ Database objects are the corporate memory of the enterprise. All the rest is anecdotes.

¹ The internal specifications of a database object are independent of its implementation. Because database object specifications are ANSI SQL1999 standard, different SQL1999 DBMS vendors (here "DB" stands for database object) are free to implement the ANSI SQL1999 specifications as they like just so long as two conditions are true: the database object specifications are portable from one SQL1999/DBMS to another, and the behavior of the same database object is same, from the user point of view, even though the database object is implemented by different SQL1999/DBMS vendors. database objects range from the trivial to the complex. A *trivial* database object is: 1) is a simple data structure (a set of single value columns), 2) is instantiated through simple databases processes (INSERT, MODIFY, DELETE) that are 3) part of one encapsulating information system, and 4) takes on a minimum of two values states: null and valued. A complex database object is: 1) a complex data structure defined within a column of a table that in turn contains multiple contained segments containing single, multi-valued, groups, repeating groups, and nested repeating groups of fields, 2) in instantiated through collections of database processes that are 3) part of one or more collections of complex information systems, and 4) that take on a whole series of discrete business policy recognizable states from null to any number of discrete valued states back to a null state. In short, the full life cycle of a business resource (employee, contract, asset, etc.).



Data is Executed Policy, A Case Study

Policies and procedures, that is, database object classes, bring order, consistency, and predictability. The larger the enterprise, the greater the dependence on policies and procedures. Data is the evidence of policy execution. An employee's record is proof that policies have been carried out. Procedures are the techniques, methods, or processes by which policies are carried out. If an enterprise has the policy is to be profitable, then its balance statement, produced by processing all the general and subsidiary journals is the measure of adherence to the policy. If policy is met the enterprise must be profitable.

The procedures are named, and their data actions are associated with specific subsets of the named data structure. The names of the procedure sets represent data structure transformations from one recognizable state to another. Each state represents a determined value set within the business. Procedure examples include: establishing an employee requisition, accomplishing employee hiring, and performing employee assignment.

Enterprise database is an organizational operating condition in which there is both defined policy coherence and integrity as well as consistency in policy transformations throughout the enterprise irrespective of functional and organizational style and irrespective of policy transformation technology (that is, computers, operating systems, programming languages, and database management systems).

Organizations not pursuing database object specification, implementation, and management information system (MIS) evolution through database objects will never achieve enterprise database. Rather, they will be left with complicated, redundant MIS specifications, expensive MIS implementation and inconsistent difficult MIS operation, evolution and maintenance.

Enterprise database is the expression, population, use, and manipulation of all database objects. Enterprise database begins first with quality database object classes founded upon the policies and procedures surrounding their specification, implementation and evolution. Value proceeds not first from the database object, but from the database object class. The consequence of quality database object classes are "real" database objects. The information technology assets of the enterprise are both its database objects and its enterprise database object classes. If only database objects are valued, then only musical notes would be needed for a great symphonic score. Performances are differentiated however, from the grade-school band to first-rate orchestra because of the musicians' talent that is coupled with the quality of the orchestra director's interpretation of the score's rhythms, tempo, articulation, and dynamics.

The remainder of this paper presents the results of a consulting assignment over the last nine months during which the principles set down above "came home to roost."



2 Case Study

The purpose of the case study is to show that if the underlying database design and the attendant software system's infrastructure are not synchronized with the policy of the enterprise, then the enterprise operates at best in a suboptimal fashion. What is presented is the nature of the enterprise, the existing database and information system's environment and consequences of that environment as of Spring 2000, and then the work that has been accomplished in just the next nine months, that is, through December 2000.

2.1 The Case Study's Enterprise

The case study's organization is a national association. The staff is small, less than 30. The members of the associations are corporate entities, government agencies, and universities. Corporate entities range from a single person to multi-billion dollar companies.

The work of the association is accomplished through committees. There are Management Committees, Technical Committees, and Technical Groups within technical committees. All committees are managed through officers. In all there are about 100 such committees. Some committees have less than 10 members and other committees have over 150 members. The quantity of company-members is about 1000. The quantity of individuals from all the company-members is about 2500.

Committees meet about six times per year throughout the United States. Each meeting range from a few days each to a week. Committees are required to report all activities to the Association. That is, attendance, ballots, votes, project progress, technical paper logs, and the like. The association is required to collect and report this data in order to support proper project progress and final product process auditing.

Committees work on projects. Projects are progressed from inception through completion through ballots at certain stages of the project's life. Work on the projects is accomplished through technical papers that contribute to or change the document that ultimately represents the result of the project. Technical papers, which are all public, formally prepared, and numbered, are presented, modified, and finally either accepted or rejected through formal votes. Some votes (that is, ballots) are subject to reconsideration ballots. There are about 600 projects across all the committees, and yearly, there are about with about 20,000 technical papers. Once projects are complete, the entire process is audited by an outside organization for compliance with established policies and procedures.

Committees accomplish their work at meetings that are called, scheduled, and are formally conducted. Attendance is taken and the business of the meeting is conducted through Robert's Rules of Order. Every meeting is formally documented through minutes. Committee



votes can be cast only by members in good standing. A member is in good standing because of promptly paid dues, regular attendance of meetings, and regular voting on all formal ballots.

The funding of the association is through membership dues. Dues are set mainly by the type of committee participation. That is, Management Committee, Technical Committees, and Technical Groups. Some classes of membership fees are based on corporate size. Larger corporations pay more for their memberships than do small corporations. Company members subscribe to products on an annual basis. Products include committee memberships, special subscriptions to committee proceedings, mailing services, and the like. Multiple persons within the member company can subscribe to different products and are charged different prices depending on their membership classification. Product subscriptions can be transferred and deactivated. Member participation classifications can be changed. Many of these product and membership classification changes result in invoice events. All invoice events must be tracked in support of good accounting practice.

2.2 The First Attempt

In large measure the enterprise had no experience with data processing other than office suites. The enterprise is first and remains a professional association. Any data processing experience was acquired through courses from the University of Hard Knocks.

For several years prior to the Fall of 1999, the association had constructed a number of different dBase style files. These files consisted mainly of members lists with names and addresses, invoice lists, and the like. There was no overall architecture or design. There was no formal project established, no methodology, and most importantly, no knowledgeable oversight review board. There was no formal requirements document, no database design, no plan to reduce or control redundancy, and no strategy to manage the interfile data relationships. In short, no system's architecture of any kind.

Starting in the Fall of 1999, there was a desire to have an Internet presence. A contractor was hired that accomplished a design founded on the existing set of files. The design of the various files was transformed into a MS/SQL schema. No database design document, other than table and column listing was however created, reviewed, or maintained. There was no overall system architecture, no process, and no business scenarios. The files were converted and the data was loaded. The contractor then built a web-based front end that was to service the needs of members, that is, membership lists, balloting, and financial status.

Still, however, there was no requirements document, architecture, project plan, and the like. The reason all these things were missing was simple. The association's staff never knew these were needed and the Management Committee never demanded them. The association staff were not IT professionals. Rather, they are Association professionals. When progress of the IT



efforts were delivered to the Management Committee, there was never enough time for formal reviews on their very crowded agenda. The Management Committee apparently didn't know the right questions to ask and, if the right questions had been asked, the Association staff wouldn't have had a clue what the question meant nor how to answer them.

Something of great value was however accomplished. Over the about 2000 staff hours expended there was a lot of data entered, it was relatively high quality, and it was able to be used during the data conversion effort. Additionally, the key association staff member assigned to the effort was acquiring a deeper understanding of data, relationships among data, and an awareness that all was not well in Oz.

By April 2000, about two years after the "real" work had begun, the Management Committee finally recognized that in the face of repeated presentations of non-completion of a "system" over the period of about 18 months, that something dramatic had to happen, or the effort had to be cancelled altogether. Whitemarsh, an association company-member for over 22+ years was called by the Association and was asked to review the situation and make recommendations. Whitemarsh brought extensive functional knowledge and over 30 years of database information systems design, development and implementation experience.

2.3 The Assessment

The first document requested was database's design. It did not exist. Asked for then was the requirements document. Then the system's architecture document. Then the process model document. Then the business scenarios document. Then, finally, the project plan was requested. Nothing existed. The only things that really exist were the collection of web-based data entry programs that collected and stored data into the database, and the MS/SQL database schema. From a requirements perspective there was nothing. From a comprehensive database perspective there was nothing. From a comprehensive systems perspective there was nothing.

A rudimentary business scenario model was created from the hierarchical organization of the web screens. Because there was a MS/SQL DBMS engine running the set of tables, the SQL schema, did however exist and could be printed from the MS/SQL engine. The database design was then derived. A quick analysis showed that neither candidate keys or any form of referential integrity existed.

Since the database's design represents the maximum boundary of the data that a system can collect, interrelate, and report, the database's schema simply represents the definition of the association's "implemented" policy. From the design it is easy to then determine what the database can and cannot support. That is, what proof of policy executions the enterprise is able to collect and upon which they are be able to report. Figure 1, at the end of this paper, depicts the database design. In this diagram, one-to-many relationships are depicted through an arrow.



Data is Executed Policy, A Case Study

The following is a brief list of the existing enterprise policies that were in force at the time the project started and that were not able to be supported by the database's design.

- ! Ballot's have questions with multiple answers that are voted
- ! Ballots are subject to reconsideration ballots that may result in different answers to the same questions. Votes have to be tracked by organization, committee, and representative over the lifetime of a project.
- ! Committee offices can be vacant and have to be tracked
- ! Committee office applications have to be tracked
- ! Different membership classes within and between committees cause different fee structures to be active
- ! Key statuses of members for dues, voting and attendance have to be tracked
- ! Member companies can merge, split, etc., and attendance, votes, invoices and fees paid have to be tracked across these corporate changes
- ! Payments need to be received and tracked with respect to invoice items
- ! Products can be subscribed by representatives and the subscriptions can be transferred to others within the member company
- ! Products of the same name have multiple prices within the same committee depending on corporate size
- ! Products exist generically. Subsets of products are allocated to committees, and subsets of these can be allocated to organizations within committees.
- ! Representatives can represent multiple organizations on the same or different committees

Given that data is executed policy, and given that data, not properly defined within a database, can not be collected in support of enterprise polices, then essentially, those policies can not be proven to either exist or be proven to have been properly executed through the evidence of



collected and reportable data. One could therefore assert that unprovable policies essentially do not exist. *“In database, if you cannot see it, it’s not there to be seen.”*

2.3 The Plan

After the assessment was presented, the association’s staff asked for recommendations. Four tasks were suggested.

- ! Create a system architecture document that acknowledged the need for a backroom system, a web presence for data from the backroom systems’s database, and a committee’s administrative-officer system to accomplish off-line committee business in support of the various committee meetings that occur throughout the United States, Canada, and cities in some foreign countries.
- ! Create a complete database design that reflects the known policies of the association.
- ! Create a complete business scenario model that demonstrates that the business of the association can be conducted through the new database design.
- ! Create a running prototype of the backroom system to show conclusively that policy is specified, data can be collected, work can be accomplished, and proof can be tendered of policy execution.

2.4 The Prototype Phase Results

The four task-based components of the plan were agreed to by the association at the end of the first week’s assessment. A two staff months estimate for this prototype phase was provided. The four tasks were completed and delivered to the client at the end of the first week of June. The documents comprised about 50 pages for the system architecture and business scenarios. The database was fully defined, and included all referential integrity. The quantity of tables grew from 26 to 47. Some tables were kept, many more were added, and quite a few were recast. All tables were bound together through policy-based relationships.

The backroom system was generated from the database design and then “electronically tuned” to meet the needs of the business scenarios. The prototype, created through the use of Clarion for Windows (www.softvelocity.com) resulted in a system that contained about 150



“programs.” A “program” in Clarion’s terms is a procedure that is activated from a menu and that results in a screen with one or more browse windows that either presents data or accomplishes a database modification. Clarion was chosen to accomplish the prototype because it enabled success by allowing maximum time for requirements, database design, and business scenario building. Once these were built, they were fed into Clarion and a working system was generated. Actually, the prototype was built within the first three weeks and used as the demonstration mechanism for the unfolding requirements and the evolving database design and business scenarios. This iterative process (requirements, design, generation, demonstration, and feedback) continued through about 5 cycles during this two month period.

By and large, all the missing functionality cited above was remedied except for two major areas: projects and technical documents. These were identified as outside the scope of “Version 1” of the new system. The critical interface tables to these two areas were however designed so it is believed that these functional areas can be added without any major changes in the database’s architecture.

Once the system architecture, database design, business scenarios, and backroom system were presented to the Association the task was considered complete. The Association had a valid and demonstrable version of the functionality that formed the basis of their policy-based requirements. It was up to the Association to select an implementation contractor and an implementation environment.

2.4 The Implementation Phase

Given the success of the prototype phase, it should come as no surprise that the effort was continued. Clarion was selected as the implementation platform and MS/SQL was chosen as the DBMS engine to manage the database. The implementation phase consisted of the following tasks:

- ! Make database design and backroom system modifications throughout the implementation phase as policy refinements dictated.
- ! Build in the necessary processes within the backroom system to accomplish a number of processes that required “real” programming.
- ! Provide database administrator services to convert the existing Clarion database design to a MS/SQL database design and to ensure that the Clarion backroom system could properly communicate with MS/SQL.



- ! Accomplish data conversion from the existing database to the new design.
- ! Create standard reports using Crystal Reports.

These tasks necessitated enlisting the services of a MS/SQL specialist. One was available on a half-time basis. These five tasks were completed in six more months, requiring about 9 staff months. Tasks 1 and 2 required full time effort, and tasks 3, 4, and 5 required about half time effort. The final Version 1 database design of the system is presented in Figure 2. During these six months, the database design grew by an additional 13 tables to a total of 60. The count of “programs” grew from about 150 to about 230.

The most significant amount of staff time was expended in the creation of the procedures that carried out all the embedded processes for new member processing, membership changes, membership jeopardy assessments for attendance, dues payments, invoicing, payments and the like. These embedded routines ranged in size from half a page to twenty pages of 4GL code.

The final set of metrics for Version 1 of the system are:

- ! 60 tables
- ! 250 “programs”
- ! 90K lines of code
- ! 1 staff year

Now that Version 1 of the backroom system is complete, the re-implementation of the web interface can be accomplished in earnest. That is being done by a different contractor and is proceeding completely from SQL views rather than base tables.

3. Lessons Learned

The following lessons “hopefully” were learned:

- ! Good intentions doth not a system make.
- ! Never attempt a non-trivial database application without thorough training, a high-quality database project methodology, and software tools like Clarion for Windows. Given the traditional metric of 2 staff weeks per “program,” Clarion for Windows cut the staff time from about 500 staff weeks to about 32 staff weeks. This was not a “RAD--do it fast make it cheap” effort, however, because all the



critical phases and steps from a well engineered database project methodology were accomplished.

- ! It was absolutely essential that a functional expert be *the* key person in the prototyping and implementation phase. Whitemarsh's 22+ years of functional expertise as a company-member of the association, and its 30+ years of experience in database projects, were significant contributors to the speed of requirements specification, iterative prototyping, and severely reduced re-work.
- ! Clarion for Windows matched the speed of specification and prototyping with its speed in system generation, zero-error code, and the ability to encapsulate embedded process logic without ever compromising regeneration.
- ! If the database design does not accurately reflect fully discovered and articulated policy then any systems effort will fail. This was proven during the first attempt (see Section 2.2 The First Attempt)
- ! It is essential that there be a knowledgeable Management Committee that makes the time and has the knowledge to accurately assess the ability of the staff to get a job done. The association's staff exercised their best efforts given the knowledge and skills they possessed.
- ! An accurate articulation of policy requires systems prototyping because that gives evidence as to whether the execution of the policy can result in the data necessary to prove the policy's execution. During the first attempt, a "system" resulted, but it could not satisfy the key policies that were in-force within the association. If the database design does not reflect the complete exposition of the policy then failure is certain.
- ! An iterative process that embodies requirements, design, generation, demonstration, and feedback is essential, BUT it is NOT a replacement for a high-quality thoroughly engineered database project methodology. Both new policies and policy refinements were discovered during the iterative process. There were two reasons why such discoveries were not fatal: very high quality third-normal form database design, and Clarion for Windows.
- ! A valid set of test data is essential for the comprehensive testing of the information system. Comprehensive system-level testing was not able to be

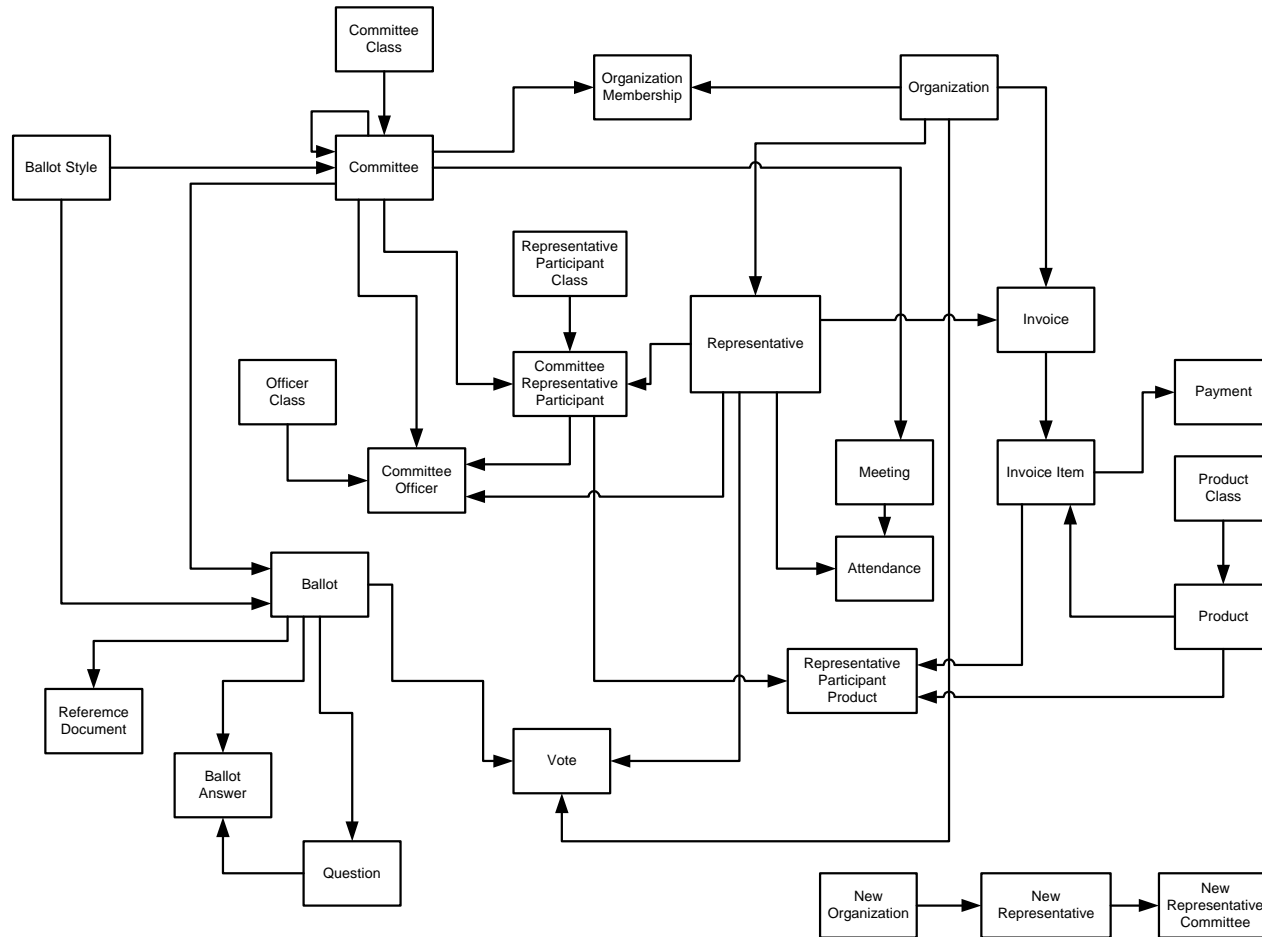


performed until the last month of the effort. During that time, “interesting discoveries” were made about the DBMS, MS/SQL. These “interesting discoveries” required slight modifications to the way the Clarion programs interacted with the DBMS. While all the “interesting discoveries” were, in the end, minor, they did cause “pain” during this highly compressed schedule.

In the end, because great attention was paid to the certain knowledge that the database’s definition had to comprehensively reflect the complete policy of the enterprise in order for the effort to succeed, three times more functionality than was contained in the first attempt was able to be delivered. And, because of Clarion for Windows, the increased functionality occurred at dramatically lesser costs than would have been expected.



Data is Executed Policy, A Case Study



Data is Executed Policy, A Case Study

