Modernizing Corporate Financial Reporting Systems

Brian Goff, M.Eng., MBA

Introduction

Historically, corporations with a large number of subsidiary holdings, have had a difficult time preparing consolidated financial reports. Constant changes in GAAP (Generally Accepted Accounting Principals) reporting requirements, including several major changes introduced in the late 1990's, further strain this already difficult process. These companies share a common goal in their efforts to find ways to reduce the monthly accounting close cycle. Even today it is not uncommon for a typical financial close process to take up to twelve days, down from as long as 45 days. As industry trends toward real-time accounting, it becomes ever more important to streamline and automate the process.

In most organizations, the Corporate Comptrollers Accounting and Administration department performs the financial reporting process. The general ledger (G/L) system will be either a homegrown system (often, originally based on mainframe VSAM files), or a packaged solution like SAP, Oracle Financials, JD Edwards, etc. These systems will also contain a

number of utilities and report writers which may have been added over the years, all of which play together in a somewhat confusing manner.

Several years ago, I had the opportunity to architect a modern financial close platform for one of the insurance industries largest companies. This project had two primary goals:

1. To develop an automated mechanism for ongoing production of GAAP-basis reports for external publication of several consolidated views of the company (these views were based on legal entity, product and accounting perspectives).
2. Produce GAAP based reports at a business unit level to support internal corporate management purposes.

On this project, the client had an older mainframe G/L system which was in the process of being migrated to a new client/server based architecture. The reporting process had to manage the consolidation of over

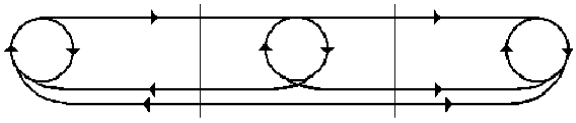
- 260 independent legal entities
- 830 individual products
- 9700 accounts (including those used for Statutory and GAAP reporting)

The Financial Reporting Process

There are three distinct phases of activities in the financial reporting process that need to be addressed:

- Data Preparation
- Analysis
- Report Generation

These phases are dependent and, in fact, need to be approached in an iterative fashion. Figure 1 illustrates the iterative nature of the financial reporting process.



Preparatory Tasks	**Analytic Tasks**	**Reporting Tasks**
G/L system reports and downloads. Data cleansing and update processes	Report line identification (i.e. Pro Formas)	Consolidation
Subsidiary and other consolidation data imports	Chart of Accounts (COA) mapping, Subsidiary work package mapping	Report production
Additional financial data downloads	Entity/Subsidiary relationships	Reconciliation
Data formatting and tracking processes	Product/BU relationships	Report maintenance
	Elimination entry identification	Journal entries (Top side adjs.)
		Elimination entries
Security, Audit and Control processes	Security, Audit and Control processes	Security, Audit and Control processes

Figure 1 - Iterative Nature of Reporting Process

In the preparatory phase, our team worked with financial experts, analysts and technical people to gather and evaluate data, and understand the initial data relationships. In the analytic phase, more detailed information about the organization structure, products, charts of accounts and supporting systems was captured and modeled in the database. Finally, in the reporting phase, reports were generated and aggregate data analyzed. At any point in this process new information that was uncovered required us to revisit previous steps of data gathering and analysis.

Approach

This client had already invested in Hyperion's new (ed. at the time) Enterprise financial consolidation tool. However, it quickly became clear that there were significant challenges in using the tool to model an organization of such a large size. For example, there were "hard" systems constraints (i.e. < 16000 names) that limit the amount of data that can be entered, and keeping track of all the modeled entities using Hyperion's proprietary language and logic was going to be tough.

One of the major problems experienced in earlier Hyperion implementations at this company stemmed from the naming convention that was adopted to meet the multiple reporting requirements. For example, the reporting line in the financial statements on which an account or portion of an account appears is based on a combination of factors: business

groups, products, dollar types, and more. Generally, this combination of factors will force the Hyperion developer to create a new "bucket" (i.e. Name) in a Hyperion consolidation structure. Each bucket/name retains that portion of the account balance that pertains to the respective financial statement line. For each new bucket, the logic that supports the consolidation of an account's balance must be defined and must reside in a logic file. As the complexity of a consolidation structure grows, the logic to support it becomes cumbersome and convoluted, and very difficult to maintain.

Our overall design approach overcame many of the problems experienced in developing the Hyperion environment for our multi-faceted reporting requirements. This improved approach, conceived by the Financial Reporting System development team, eliminated the need to code the extensive logic in Hyperion that parsed balances into more than one rollup from "jointly owned" names (i.e., a subsidiary owned by multiple business groups). We developed an application that acts like a shell around Hyperion to automatically create unique names and all the Hyperion logic for the respective rollup or view. By using this custom application to manage name structures and data, users can allocate and roll up specific balances based on straightforward consolidation logic.

The balances loaded from the G/L system into a particular name (a name is associated with a either a legal entity or business unit, and an account from the Chart of Accounts) are calculated and loaded so that it is correct for the intended view. This is done in the Application Shell where the underlying data relationships are managed with a sophisticated account

mapping graphical facility that supports complex account mapping relationships.

Using a Sybase database engine and a GUI built in Visual Basic, we used the flexibility and power of the relational database engine to make an environment that is more suitable for generating data than the multidimensional database environment in Hyperion. (Hyperion uses a database technology called a multidimensional db (MDD). MDD databases are optimized for cross-tab queries but they require very specific data representations before they can perform well.) Within the Application Shell, the user can access a variety of tools (screens and utilities) that assist in ensuring referential and data integrity, as well as provide control over the entire process. It also permits users to enter the Hyperion environment at any time in order to review financial and analytic reports for external GAAP and, in the future, Statutory and Management reporting.

System Architecture

Figure 2 illustrates how the Application Shell surrounds Hyperion. The figure also shows how Hyperion is directly accessible to the user. Data sets obtained from the G/L system and from the subsidiaries themselves were loaded into the Application Shell and were traceable throughout the process. Import functions support transparent and automatic data loading (where possible).

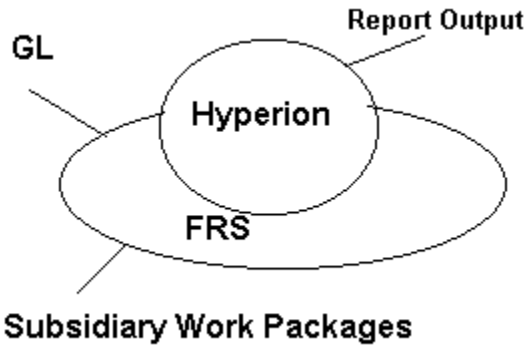


Figure 2 - Overall System Architecture

Hyperion provides “hot” links to the Hyperion Report Writer as well as other reporting tools such as Excel and Lotus. Links to other systems are possible by exploiting the openness of the system.

The architecture of the Financial Reporting System is designed to be as flexible and easy to use as possible, while supporting the level of functionality needed to support the tasks of assimilation, preparation, loading and reporting of GAAP and STAT financial data. The architecture of the system is best described as a “loosely coupled” collection of subsystems that are designed around various data, or subject, areas. The relationship of the various subsystems is illustrated in Figure 3 below:

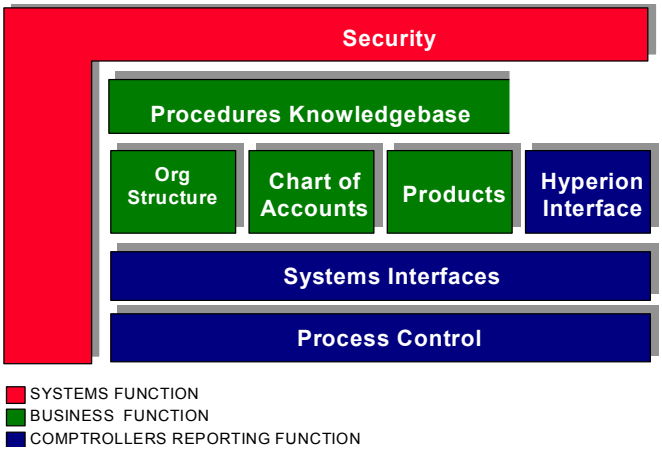


Figure 1 - FRS Subsystem Configuration

Summary

The mechanics of the financial reporting process begin to get complicated when the results of many entities are combined or consolidated into comprehensive reports. The complexity of the process depends on a variety of factors including

- The size of the chart of accounts
- The complexity of the financial statements
- Differences between subsidiary and parent charts of accounts
- Regulatory requirements
- The rollup of the Chart Of Accounts to the financial reports may be convoluted, or more commonly, vague and not well understood
- Subsidiaries may capture varying degrees of accounting detail

- Subsidiaries may report their results at different times
- Incompatibility between electronic reporting systems
- Conflicting management performance objectives may bias results.

Our design philosophy dictated the use of the right tool for the right job. By leveraging the "openness" of today's software products, especially in the Microsoft Windows environment, we maximized the value obtained from each tool while minimizing any exposure to their respective drawbacks. For example, relatively mature relational database technologies offer database engines that have been optimized for high-performance queries which gives users a great deal of flexibility and power in managing large data sets. Similarly, software designed for financial consolidation, drill-down and rollup reporting, data pivoting, and trend analysis achieves high degrees of performance in these areas.

In summary, we successfully met our goals by realizing the synergy's offered by using different tools in conjunction with one another, and employing life cycle development techniques as CMM.