Very Large Calculation Systems

A Specialized Solution for the Complex Needs of Advanced Knowledge Workers

James Madison

Abstract

Motivation. Advanced calculations on large data sets provide important business insights. Such calculations must be flexible enough for the dynamic nature of advanced analytics done by actuaries and other high-skill users, yet must also leverage the power and stability of large-scale IT systems. The system design detailed in this paper balances these concerns, delivering the optimal combination of both.

Method. The design is based on a pattern the author observed repeatedly during a decade of building systems for actuaries and researchers: data is provided using traditional data warehousing, the calculations are implemented by IT in a manner that solidifies stable elements yet externalizes change-prone elements, system operation is exposed through self-service interfaces that allow users to configure and run the calculations against large data volumes, and the output is delivered by standard business intelligence tools and customized approaches.

Results. Systems with this design: optimize the users' time by moving the majority of the lower-value work to IT, have solid auditability and legal compliance, and provide high computing power and storage capacity, but do require users to give up some amount of control and flexibility.

Conclusions. Organizations should use this design to give their most advanced knowledge workers high power, localized control, and optimal efficiency.

Keywords. Data warehousing, exploratory data analysis, actuarial systems, ratemaking, modeling, simulation.

1. INTRODUCTION

Creative knowledge work and solid IT systems are necessarily at odds—the former requires high flexibility, the latter requires high stability. Reconciling their natures is difficult, so many organizations leave them separate. This can leave important knowledge work disconnected from mainstream systems and relegate mainstream systems to lower-value analysis work. Organizations seeking to maximize every analytical opportunity must bridge this gap by maximizing the efficiency of each area, then fully integrating them. This paper describes how to do it.

Flexibility is a dominant characteristic of creative knowledge work. Actuaries may operate from a strong mathematical foundation, but new and emerging opportunities start as little more than vague ideas. Turning ideas into actionable information requires looking at data—often lots of it—manipulating it, observing the outcome, getting more data, manipulating it more, observing it again, and so on through many iterations, often with several serendipitous epiphanies along the way.

Stability is a dominant characteristic of IT systems. During development, systems must be built to a fixed timeline and budget, multiple IT skill sets must converge on a single outcome, developers must learn business logic, and changes to code once it is written can have large propagation effects. During operations, systems must meet service level agreements (SLAs), be supported by IT staff of lesser skill, exist in the corporate systems environment for many years, rarely crash, and have their changes coordinated with multiple business units.

Little about these two worlds works well together. The actuary rightly asserts, "I don't know what I want," and IT says, "We can only build it if you specify it." When something finally does get produced, the actuary sees it for the first time and immediately recognizes a way to do it better. The change may take only minutes to observe, but then requires hours or days to explain to IT, and weeks or months to implement. If getting it through IT is painful enough, the actuary may just solve the problem in her favorite desktop analytical tool where the valuable logic stays until the day she leaves—with no one else knowing how it works, thus leaving the organization in a precarious position.

Faced with this conflict, most organizations silo the two needs. Not intentionally, but de facto. Actuaries do some of their most important work in ways that few others understand. IT builds comparatively low-value systems that are quite useful for standard business intelligence (BI) but miss out on the greatest analytical opportunities. The solution is to build a middle ground that serves the needs of both areas, bringing together the value that each provides while also maximizing the

efficiency of each area individually. When done properly, organizations can be confident that the full spectrum of analytic work is efficiently supported in their organization—from empowering the most creative analytic work, to codifying all the best ideas in long-term IT systems that protect and deliver the organization's most critical competitive analytical knowledge.

1.1 Research Context

Industry literature elaborates extensively on systems and data that are useful to the majority of the business community. These are shown in yellow in Figure 1.

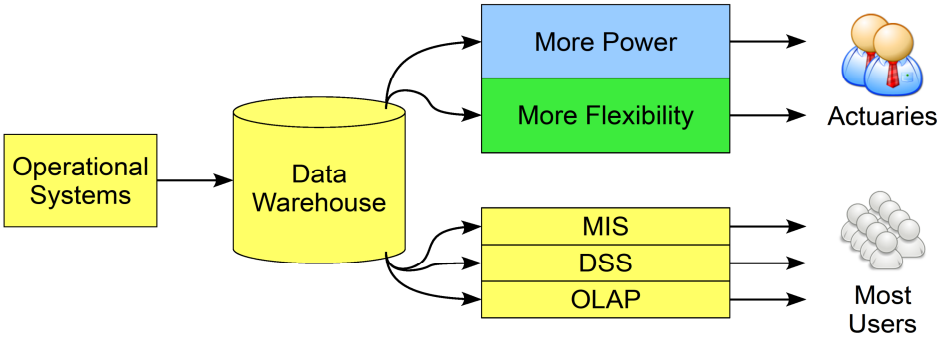

Figure 1 – Standard analytical systems serve most users—actuaries need more.

Operational systems, such as those in call centers and on consumer Web sites, generate transactional data that flows into a data warehouse (DW). The DW integrates the data into various standard analytical systems including management information systems (MIS) for executives, decision support systems (DSS) for intermediate managers, and on-line analytical processing (OLAP) systems for analysts.

Trouble is, a small but important minority of the business community needs more from their systems and data than standard analytical systems can deliver. Actuaries, statisticians, engineers, researchers and those in similar roles need more than the reporting or drilling that standard analytical systems provide. They need systems with the high flexibility required by the creative nature of advanced knowledge work. They need systems that perform advanced calculations—statistical analysis, simulations, data mining, conditional logic, predictive analytics, optimization modeling, emerging algorithms based on the cutting-edge of industry knowledge, and the many tried-and-true insurance algorithms that are inherently advanced such as rerating, loss development, risk analysis, reserving, and indications. They need to perform these calculations against large amounts of data—terabytes of data, collected from many sources around the organization, enhanced with external data,

preserved across years, even decades, with potentially many intermediate what-if versions. They need significant system resources—the raw computing power necessary to run all these calculations against all this data, yet still have answers come back at the speed of thought, lest important insights get lost while waiting for information to return. These needs demand a level of power, arbitrarily defined system behavior, and data use that is difficult to predict in advance. These needs are shown categorically in blue and green in Figure 1 and are the main focus of this paper.

MIS, DSS, OLAP, and similar mainstream analytical systems simply cannot handle these extreme needs for high-power and high-flexibility, nor were they ever intended to. Specialized vendor tools may attempt to address such needs, but vendor tools necessarily make generic assumptions that serve a wide customer base, thus making it difficult for any one customer to incorporate the uniqueness that creates competitive advantage—the very thing that most high-end analysis is intended to find!

1.2 Objective

The very large calculation system (VLCS) handles these extreme needs. The VLCS is a system design pattern that organizations can follow to create systems that provide calculation-intensive processing, large-volume data handling, and high user flexibility, while also addressing corporate IT processes, legal compliance, and similar non-technical constraints.

The objective of this paper is to address the major components of the VLCS as they are seen from the view of both actuarial and IT. The actuarial view details the functionality provided, what is possible with good system engineering, and what actuarial can reasonably demand from IT. The IT view elaborates on important back-end details, but in language meant to be understandable by technically savvy actuaries in joint conversations with IT.

1.2.1 Points of Clarity Before Starting

Nearly every aspect of the VLCS design already exists in some form in most organizations and has been written about somewhere in industry literature—this is one of its greatest strengths—no single aspect should be new or peculiar to actuaries or IT professionals with a few years of experience in a complex environment. The unique value presented here is that of providing a unified architecture, common language, and solid roadmap that all stakeholders can use to explicitly define and design such systems, and organize and justify the projects that build them.

Actuaries are not the only ones who need VLCSs, so non-actuaries are invited to interpret all references to *actuaries* and the *actuarial department* as their own role and group. Likewise, not all actuarial work demands the extreme power of a VLCS, so *actuarial work* as it is used here means the high end of the actuarial work continuum, where a VLCS might be applicable.

In any sufficiently advanced computing system, many system components interact across multiple layers of hardware, software, networking, and overall system design to deliver the simple outcome seen on the screen of the user. This paper uses the term *tool* to mean the simple thing the user sees on the screen and uses to get his job done. See the Definitions section for more.

2. TOP-LEVEL ARCHITECTURE

The top-level architecture of the VLCS is shown in Figure 2.

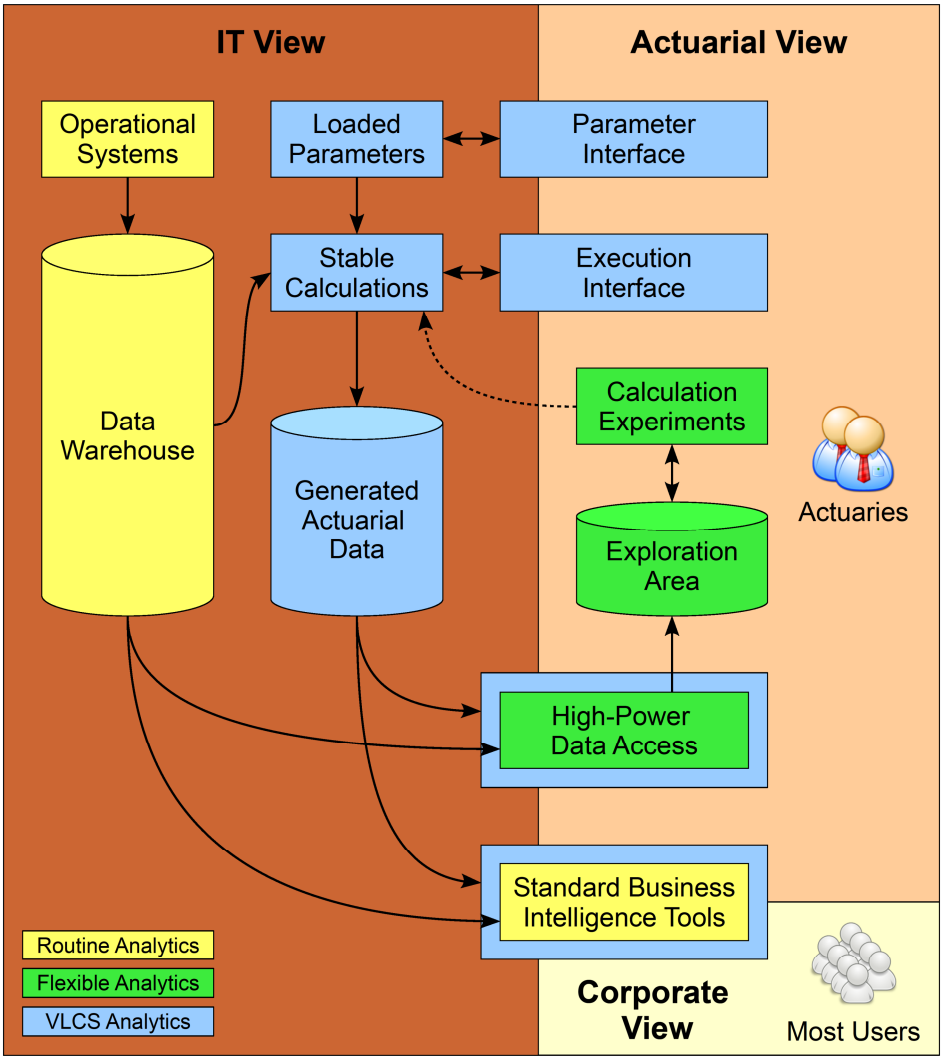

Figure 2 – Top-level architecture of the VLCS

The diagram reads as follows: *Operational systems* feed into a *data warehouse* that cleanses and integrates the data and preserves its history. *Standard business intelligence tools* deliver the data to *most users*, meeting most of their analytical needs most of the time. *Actuaries* tend to have more advanced needs, so actuaries access the data through *high-power data access* in addition to the standard tools. Actuaries keep this data and various working data sets in an *exploration area* that is fairly large and minimally governed. Actuaries perform data manipulations, what-if analyses, data mining, and other *calculation experiments* requiring a high level of flexibility. Some fraction of that work eventually yields important and clearly defined insights that are converted to *stable calculations* that IT builds into the organization's mainstream systems. Stable as they are, such calculations still vary across such things

as time, geography, customer traits, line of business, product, and similar grouping criteria, so a *parameter interface* is provided to give actuaries direct control over the calculations with no IT involvement. Running the system is done either by IT or directly by actuaries using an *execution interface*. The execution of the stable calculations using the *loaded parameters* results in *generated actuarial data*. This new data is fed back into the same flows as the regular DW data. Feeding it into the standard BI tools gives the larger business community well-defined and governed access to actuarial's most advanced calculations. Feeding it into the high-power data access gives actuarial a closed loop that allows large-scale what-if analysis using the full power and deep data of the DW platform.

Systems with this architecture can directly address the most data- and calculation-intensive questions a business can ask. Among the most advanced applications personally encountered by the author are rerated earned premium at present rates, loss development, reserving, geographic risk concentration, and rate indications. Other good candidates include any business question requiring years or decades of fine-grain data to be manipulated with highly detailed and varying parameters.

3. ACTUARIAL VIEW

The components of the actuarial view fall into three categories: 1) Routine analytic components: those that IT delivers to any team with repeatable work needing routine analytics. 2) Flexible analytic components: those delivered to any team with open-ended work needing flexible analytics. 3) VLCS components: those delivered as part of controlling the VLCS and utilizing its output. These categories are shown in yellow, green, and blue respectively in the top-level architecture.

3.1 Routine Analytic Components

Before attempting a solution as advanced as a VLCS, an organization needs to have succeeded at the basics. This includes, at a minimum, having a DW and using standard BI tools. The DW is required because it is the source of data that goes into the VLCS, and because it stores the VLCS output for integration with the larger organization. The BI tools are required because they answer basic analytical questions that establish the context for VLCS answers, and because they are the channel for delivery to the larger organization. DW and BI tools are covered at length in industry literature, both in general and with an actuarial focus[1], so they are reviewed here only enough to show their important foundational effect as it relates to the VLCS design.

3.1.1 Data Warehouse

A DW is a large repository of data that is historical, integrated, granular, and subject-oriented[2], referred to here as HIGSO form. Before building a VLCS, ensure that a large data repository can be found that has as many HIGSO characteristics as possible: many years of history (at least five years typically, up to 20 or more for asbestos, agent orange, mold, and other long-tail concerns), integration of many sources (policy writing system, claims system, etc.), deep granularity (vehicle identification number, not just year, make, and model; 9-digit zip, not just 5-digit zip; latitude/longitude, not just 9-digit zip; etc.), and diverse subjects regardless of which source it comes from (risks, coverages, vehicles, geography, service requests, demographic characteristics, etc.). Such an environment is a prerequisite to the VLCS because it provides the deep data that the VLCS must process.

3.1.2 Standard Business Intelligence Tools

The majority of access to the DW is done by commercial BI tools. At their simplest, these tools provide predefined reports. At their most robust, they provide OLAP and basic predictive analytics. For our purposes, the robust end of this spectrum can be considered in place when users can do the following with delivered data:

1) Drill down/up – Moving to lower levels of detail and higher levels of summary. For example, state down to county, down to city; agency up to territory, up to region.
2) Drill across – Moving among different collections of metrics. For example, from premiums across to losses to find loss ratio across to expenses to find combined ratio.
3) Slice & dice – Narrowing the metrics to a specific scope of interest. For example, only the male slice of customers in the urban-and-coastal dice of geography.
4) Statistics & trends – Performing basic analytics across time. For example, show standard deviation to find catastrophic outliers and do this for the past 36 months.

Achieving these four fundamental BI activities is a function of both the DW and the BI tools, with the data model of the DW being the more important of the two. A DW that facilitates these activities is known as dimensionally modeled[3]. It is the dimensions of the data that the user is drilling, slicing, and trending. Any organization that is serious about using its DW for driving its organization with analytics must model its data dimensionally. BI tools must then present this dimensional model in its most usable form. The dimensional form has a number of variations and

names such as OLAP, star schemas, pivot tables, and cubes; in practice, these are all just variations on achieving the same four activities.

The four fundamental BI activities combined with the four HIGSO characteristics of a DW constitute the majority of analysis that most business users need to do, even actuaries. If actuaries can drill up, drill down, drill across, slice and dice, apply all major statistical functions, and trend across any amount of history, data sourcing, depth of grain, and subject orientation, how much is left? All that is left are special situations and hard cases, which together lead us down the paths of flexibility and the VLCS.

3.2 Flexible Analytic Components

The DW and BI tools together should prove to be a valuable foundation for any organization's analytical needs, but when they also prove too rigid for certain dynamic needs, more flexibility must be introduced. The components of flexibility are access more powerful than standard BI tools, a very large storage area to hold data of arbitrarily defined data sets, and enough computing power to run complex analysis.

3.2.1 High-Power Data Access

High-power access results from powerful tools and almost unlimited authority to read the data. Such tools and authority contrasts with standard BI tools, which are designed to provide ease-of-use for routine tasks, and whose authority model is one of limiting users to only well-defined and highly refined data—a good value proposition most of the time, but potentially limiting for advanced work.

From the actuaries' view, high-power access essentially means having tools that allow the direct input of SQL and the capability to run it directly on the DW platform. Direct SQL input should be in addition to simpler graphical functionality that continues to keep routine work simple. The tools are usually the easy part—most good standard BI tools have a "back door" that can be easily enabled by IT to allow arbitrary SQL to be input and run on the underlying platform, and specialized data access tools almost certainly have such behavior.

The harder part is that IT is often concerned that actuaries may consume too much system power or may misunderstand the data. The power-consumption risk exists at several places throughout the architecture and will be addressed in the *workload management* topic toward the end of the paper. The misunderstanding risk is a serious one and is best addressed with the following policy: for a fairly high percentage of the data, actuaries might be on the cutting edge of the

company's understanding of its data, thus largely on their own. For example, a company's policy writing system might be 30 years old and capture 20,000+ data elements, but the DW may only officially define and support 500 of them in full dimensionalized HIGSO form. The benefit of virtually unlimited access is that important data elements could be discovered that would never see the light of day if actuaries could not dig into them. The risk is that actuaries may find themselves weeding through useless data that could get misinterpreted or find its way into institutional reporting contexts. The tipping point between the benefit and risk cannot be predefined, but the goal is to empower actuaries with SQL against all possible data so that their judgment of what is and is not valuable can be brought to bear, rather than IT filtering such judgments via typically slow and expensive processes.

3.2.2 Exploration Area

High-power data access reaches into the DW in a read-only manner, but actuaries frequently need write access for such things as storing results when query q_{n+1} requires as input the output of q_n, developing ideas by reviewing intermediate results, and storing particularly interesting outcomes for later review. These needs are met by having a large area with write access and high freedom known in the industry as an exploration area[4] or sandbox[5]. This area is a logical concept that IT can physicalize on any major database and server product suite—from large, generic database systems like Oracle and Microsoft, to specialized players like Teradata, Netezza, and SAS. This area should be as connected as possible to the main DW, provide significant storage capacity, have a liberal usage policy, offer some level of monitoring and maintenance, but restrict institutional reporting.

The optimal connection between the exploration area and the DW is achieved by collocating them on the same hardware, thus providing the maximum data transfer rate and the sharing of common tools. If collocation is not possible, a high-speed connection must be used between the areas so that actuaries can move data from the DW to the exploration area as efficiently as possible.

The storage capacity of the exploration area must be large enough to accommodate the base data pulled from the DW as well as intermediate result sets produced by the actuaries. As a starting estimate: determine the largest DW table the actuaries will use, multiply that by two to accommodate all the small tables that will be joined to it, multiply that by three for stored intermediate results, and multiply that by the number of actuaries doing exploration work. This usually results in a financially non-trivial number; constrain it as needed based on budget realities. When allocating this space,

avoid the temptation to have shared accounts since the lack of accountability can cause things to grow out of control. Instead, allocate the space to individual accounts and have IT train the actuaries in cross-account sharing, which they should find quite straightforward once they know it.

The usage policy on the individual accounts must be extremely liberal, the only real restriction being the total storage capacity allocation. Avoid or minimize other restrictions such as limits on the number of queries, types of queries, loading of third-party data, and similar activities.

Monitoring and maintenance of the exploration area should be done by IT to help the actuarial community understand what is going on in the environment: tracking data sets by who created them, who read them, and the operations performed against them; backing them up, recovering them if something goes wrong; having the dates for all these activities. These can be very useful to actuaries and are relatively easy for IT to establish using functionality built into the platform itself. Such value-adds are one of the main reasons the organization should build a centralized exploration area, have IT manage it, and encourage actuaries to use it. More advanced monitoring in the exploration area, such as verifying the quality of any particular data set, tracking the lineage of which data sets were used to produce which other data sets, or certifying data sets for their legal compliance, cannot be done in the exploration area by using built-in platform functionality and instead require the kind of customized programming that can only be done as part of building an IT system. This more advanced monitoring would also require having to institute restrictions on what actuaries can and cannot do, which begins to undermine the primary purpose of the exploration area.

Because quality, lineage, certification, and similarly advanced monitoring is challenging, the exploration area should produce little to no institutional reporting, or if it does, it should be clearly disclaimed and done at the risk of actuarial management. Institutional reporting is reporting that goes beyond the actuarial department, including senior leadership for important decisions, other departments for their operations, government agencies such as state insurance offices, and external organizations where SOX, GLBA, HIPAA, and similar laws may apply. Given that the cost of productionalizing exploration work can be as high as nine times the cost of developing the exploration work itself[6], it can be quite tempting to do institutional reporting from the exploration area, but this is a very slippery slope that must be actively avoided. Institutional reporting that requires the output of advanced calculations strongly indicates moving to a VLCS since IT is long-accustomed to handling compliance.

3.2.3 Calculation Experiments

With access to unlimited corporate data and ample working space, actuaries can now perform calculation experiments that explore the ideas they are attempting to turn into actionable information. Such experiments necessarily require extensive querying, joining, binning, filtering, redefining, translating of values, and storing of intermediate result sets—actions that generally result in some form of computer code, the size and complexity of which can become significant. The intensity of the work and the extensiveness of the code require that IT provides sufficient computing power, the freedom to install new or specialized software, and the right to use official corporate software in a highly flexible manner.

Sufficient computing power, like high-power data access, pushes in the direction of collocation of advanced actuarial work with the main DW. The DW will generally have more power than any secondary platform that could be created. Keeping the experimental work on this platform leverages this power and prevents having to justify and maintain stand-alone hardware.

Software for calculation experiments must balance two competing forces. It must include any software that actuaries need to work efficiently, yet it should consider the possibility of promotion to an IT system in the future. The need for efficiency requires that IT policies allow software installation in individual user accounts with minimal barriers. It also requires IT to help install such software when doing so requires system-level changes that are not possible with individual user authority alone. The need for future promotion asks that actuaries consider tools used by IT as the first suite of actuarial tools. If actuaries are open to using IT-sanctioned tools from the outset, even if they may not be their favorite or their first choice, it will make promotion of the stable calculations into a formal IT system much simpler and cheaper in the future. For example, actuaries might be inclined to pull data out of the DW and into Microsoft Access to do exploration, but if they work in an organization that uses Microsoft SQL Server as the DW platform, the actuaries might find that working directly in an account on the SQL Server is not much harder than working in Access, yet it is much easier to have IT productionalize SQL Server work than Access work should the work grow beyond just exploration.

3.3 Justifying the VLCS

The DW, BI tools, and exploration area together constitute a legitimate service model from IT to business areas. It may be quite reasonable to stop at this point and declare the IT service model for data delivery complete. Stopping here could also present several challenges. These are briefly

described in the "Consider a VLCS if…" column of Table 1. These challenges are directly addressed by creating a VLCS. However, creating a VLCS presents its own challenges. These are detailed in the "Defer a VLCS if…" column of Table 1. The situation presents a classic trade-off. Determining which path is right for a particular organization, department, or team is entirely up to those involved. The decision makers in such a situation are invited to use Table 1 as a guide during the decision making process.

Criteria	Defer a VLCS if…	Consider a VLCS if…
Routine work	**Integrated** – Routine work is tightly coupled with creative knowledge work	**Compartmentalized** – Routine work can be separated and handed to IT for automation
Structural stability	**Dynamic** – The actual nature of the analysis work changes as discoveries are made	**Stable** – The nature of the work does not change, only the data and parameters going into the work
Legal compliance	**Localized** – Decisions based on analysis do not go beyond well-defined internal use	**Constrained** – Legal or internal compliance require auditability and similar standard value-adds from IT
Corporate integration	**Limited** – The users of the actuarial work are actuarial itself or a small community they can manage	**Shared** – The work of actuarial has an organizational benefit realized by connecting it to the main DW flow
Bureaucracy tolerance	**Eager** – The turnaround time required for changes is short even for fairly stable elements of the system	**Patient** – The natural latency of IT is bearable when changes are requested for components under IT control
Staff change	**Stable** – Turnover is low and advanced-but-recurring work is well understood by several actuarial staff	**Churn** – The joke "no one's quite sure how it works" is more ominous than humorous when actuaries leave
Processing intensity	**Moderate** – Processing can be done within required time windows using hardware under actuarial control	**High** – Processing intensity demands the kind of hardware that actuarial cannot support on its own
Data volume	**Low** – Exploration area flexibility is valuable enough to justify having to manage the data and code in it	**High** – Working the data in its native environment provides storage and processing efficiency worth having

Table 1 – Criteria to weigh when considering a VLCS.

3.4 VLCS Components

So that the actuarial view can be discussed in its entirety, assume that a VLCS is justified, has been built, and has been delivered according to the details discussed later in the IT view. As part of the delivery, actuaries must have interface points that give them a high level of control, and simple yet powerful access to the output for themselves and the larger organization.

3.4.1 Parameter Interface

The parameter interface parameterizes business rules and the data generation process, putting both under the direct control of actuaries. Parameters are inputs to business rules that cause outputs to vary. Standard BI tools have some level of parameterization, the VLCS has more. BI tool parameters include such things as pick lists, user-defined hierarchies, pre-defined aggregates, value banding, and other inputs that influence the output, but not the business rules nor the state of the underlying data. VLCS parameters influence output, but they also vary business rules and change the data that gets generated.

As examples: A traditional BI tool may allow actuaries to group losses by range where some range of values is classified as catastrophic, but a VLCS might allow actuaries to provide catastrophe thresholds that cause different rating variables to be used to generate rerated earned premium. A traditional BI tool may allow trending of losses by geographic characteristics, but a VLCS may allow the calculation of potential future losses based on geographic proximity to geographic risks such as fault lines or nuclear power plants as defined by actuaries. A traditional BI tool may allow analysis of losses across time, but a VLCS may generate reserves for losses projected to ultimate based on assumptions provided by actuaries.

Notice that for each of these, variations in input (e.g., catastrophe thresholds and rating variables) cause variation in business rules (e.g., which thresholds cause which rating variables to be used) or generated data (e.g., the earned premium based catastrophe-stratified rates)—variations that cannot be solved from source data alone. What constitutes a catastrophe? Which rating variables are to be used for such catastrophe thresholds? How are exposures to be rerated? What trending assumptions are to be used to define the future? How close does a location have to be to a risk to be worrisome? Which locations are considered risky sites? How conservative or liberal should the reserving policy be? What are the assumptions on ultimates? These questions have three important characteristics to observe:

- They cannot be answered solely from basic DW/BI functionality—the downstream actuaries are the authorities on the question, and there is no realistic way to operationalize this into something like a call-center question, so actuaries must have an input path for these drivers of business rules and data generation.
- They are *not* stable enough to define for a one-time IT build. As soon as actuaries see, for example, the rating changes based on changed catastrophe thresholds, they will want to change them again. What-if loops frequently become a core VLCS behavior.

- They *are* stable enough to be given some basic structure that the IT system can consume. As a basic guideline: are the parameters that vary at least stable enough to be put into a spreadsheet of a few tabs with a few columns each? If so, the VLCS can be fed this information as input.

3.4.2 Execution Interface

With parameters loaded, actuaries run the system using the execution interface. The execution interface provides both self-service invocation of the VLCS and the status of all other invocations of the VLCS, both current and historical. The invocation portion of the interface is little more than a "Run!" button that starts a run of the VLCS. More important is the information provided by the interface about the other invocations of the VLCS—information that allows actuaries to be rational about the use of system resources and to be aware of previous runs for reuse. At a minimum, the interface should show the total system load, the number of processes the VLCS is currently running, their most significant parameters, who invoked them, and the approximate completion time. The goal is to allow the users to use the VLCS to its full capacity without overwhelming the system. The assumption is that power users with enough years of experience understand the nature of sharing large systems and will do so rationally if properly informed. Just in case such an assumption of rationality is a bit too optimistic, IT can put in controls discussed later. The interface should also provide historical invocation information so that previous result sets can be reused. For example, if a state was just rerated with the latest rate tables last week, that should be shown so no one else runs it again needlessly. The parameter and execution interfaces are often provided via the same application. An example of an interface providing both parameterization and execution functionality is shown in Figure 3.

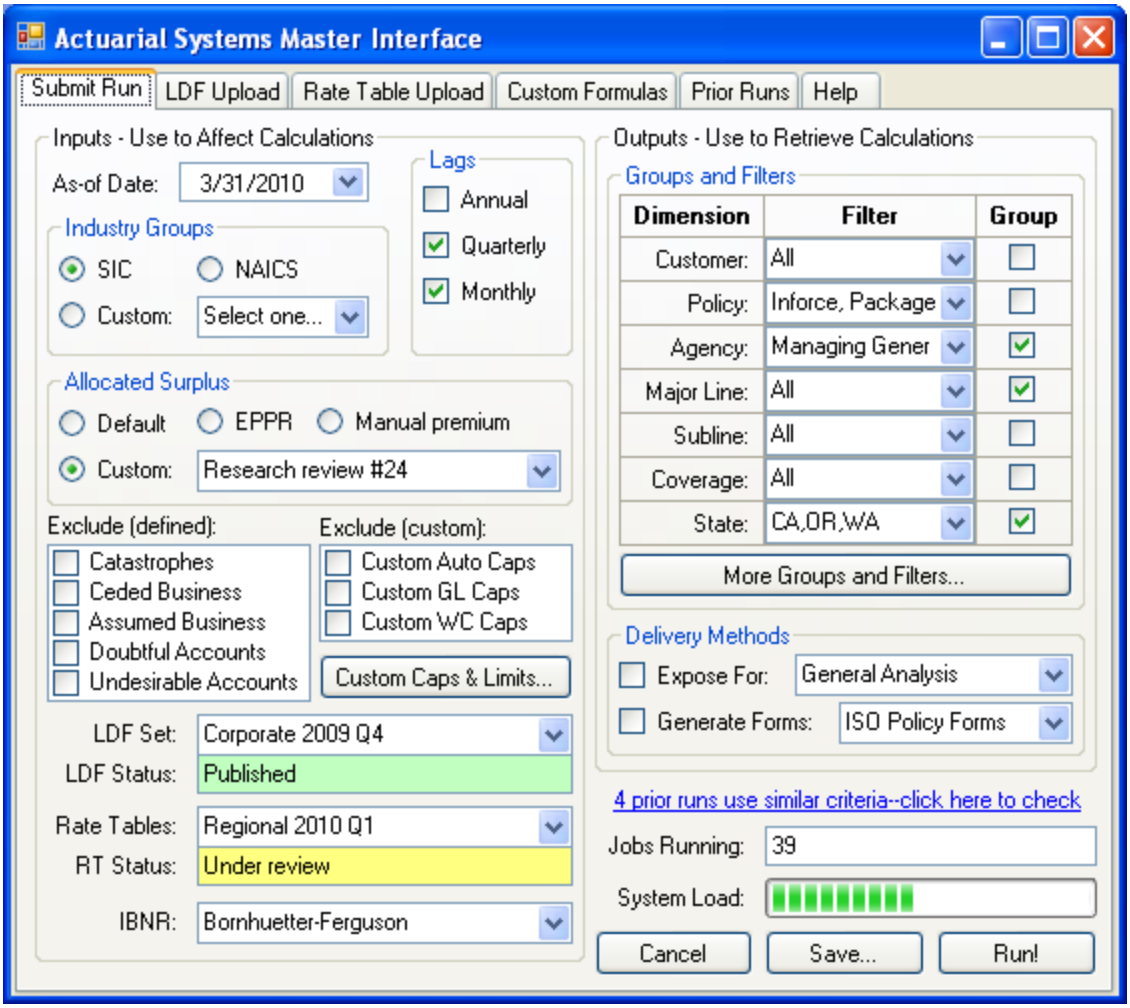

Figure 3 – An example of an interface with parameter and execution functionality.

The figure is contrived to facilitate our discussion here; an actual interface would be many screens, provide even more functionality, and be much more usable; but the figure is representative, and allows us to have a discussion that holds in reality even if the actual figure does not. Some things to observe from the figure follow. There are four major regions: items that are inputs to calculations are on the left, items that affect reporting outputs are on the top right, items that indicate system state are on the lower right, and items that have complexity unto themselves are on the tabs across the top. The calculation input items include things that will change the behavior of the system processing such as which as-of date to read from the DW; which industry grouping to use; what to include or exclude; what LDFs, rate tables, and IBNR algorithm to use; and the

certification status of items that are themselves data sets. The reporting output items include such things as how to group and filter; who to expose the output to, such as everyone for general use, internal research only, or other groups; and whether to generate pre-defined formats such as ISO forms or other corporate and industry templates. The reporting output should be somewhat limited since most general purpose reporting should go through the standard BI tools, but some customized and immediately generated predefined reports are typically useful. The system status items give a general indication of the load on the system to facilitate efficient sharing and alert the actuary if there are similar jobs worth reviewing to possibly avoid doing redundant work. The tabs provide extended interfaces for specific parameter sets of high complexity; for example, rate tables, which typically vary by state, line of business, and product, and may have many exceptions, variations, and historical versions.

3.4.3 Generated Actuarial Data

A run of the system generates actuarial data, raising the issues of storing it, certifying it, integrating it, and delivering it both to actuaries and the larger business community.

Storage initially should be within the DW. Collocation with the main DW data allows for simplified access and high-performance when compared to storage on platforms outside the DW. If the data will never undergo certification (such as with runs used entirely for what-if and similar research purposes), it should be copied to the exploration area and deleted from the DW.

Certification of the data is an organizational process involving review by key individuals or a governing body and the labeling of data according to some continuum, such as:

- Personal – the data is for one person doing what-if analysis; never to be promoted
- Uncertified – recently generated with the intent to promote
- Under review – being reviewed for promotion by the appropriate authority
- Certified – approved by the appropriate authority
- Published – sent to other organizations, often resulting in legal/archiving restrictions
- Obsolete – not used, may be a candidate for elimination or long-term archive

The certification will likely cover only a very small amount of the generated data since much of what the VLCS will be used for is what-if analysis that enhances the understanding of actuaries trying to discover something.

Integration is a standard DW practice that can be done with VLCS-generated just as it can with any data source. The integration is possible because the *process of generating* the data may require a VLCS, but *the end result* is little more than numbers that need to be drilled, sliced, diced, filtered, grouped, and treated like any other dimensional data. For example, rerated earned premium is a very difficult number to produce, and generally requires some form of a VLCS, but "rerated earned premium by agency, state, and class of vehicle, by month, trended over the last two years" is simple to provide via standard BI tools—once the rerated earned premium has been generated by the VLCS. Note carefully: collocation means putting the data in the same place physically, providing system performance and positioning it for possible integration. Integration means putting the data together logically, providing a unified business view—a proposition both more valuable and harder to do than just collocating.

Delivery of the generated data before it is certified and integrated should be through the high-power data access flow. This allows actuaries to review the data before expending organizational effort on certification, expending system effort on integration, or exposing the larger business community to immature data. Delivery of the generated data after it is certified and integrated should be through the standard BI tool flow, a simple proposition once integration has occurred.

3.5 Summary by Example: Data Mining

Data mining provides a great example of a problem space that is valuable for actuarial work[7], has product solutions with many of the characteristics of a good VLCS out-of-the-box, and yet still requires supplementation by IT to create a full VLCS solution. Data mining in its most basic form has the steps shown in Figure 4.

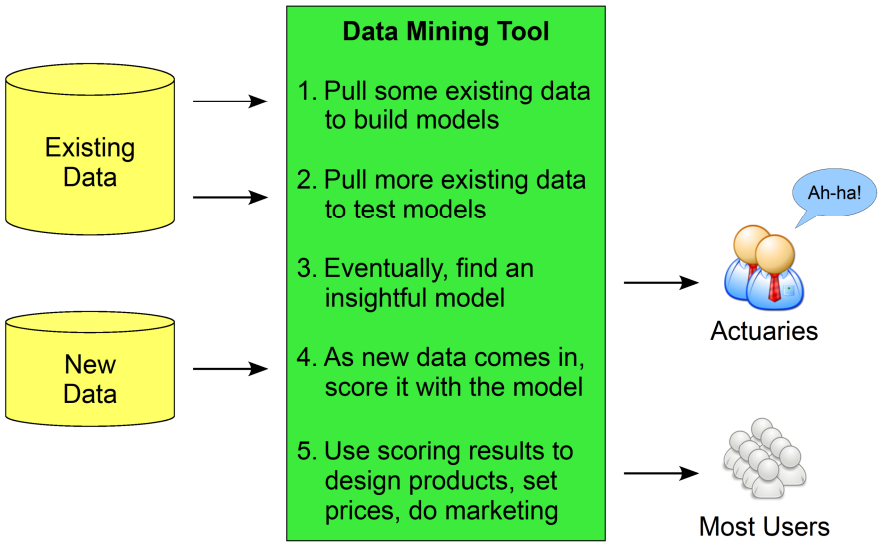

Figure 4 – Basic data mining: valuable functionality, on a small scale, managed by actuaries and business units

If the organization's goal is to have a handful of smart actuaries construct a small number of models for well-defined user communities, then Figure 4 is a cost-effective, manageable approach. The mining tool can be installed in the exploration area, the data pulled from the DW, and the deployment to users managed as part of actuarial operations. However, if the organization finds that the approach causes valuable actuaries to do low-value recurring tasks, worries the legal department due to the questionable audit trail, and creates similar forces that push from the left side of Table 1 to the right, the work can be built into a VLCS as shown in Figure 5.

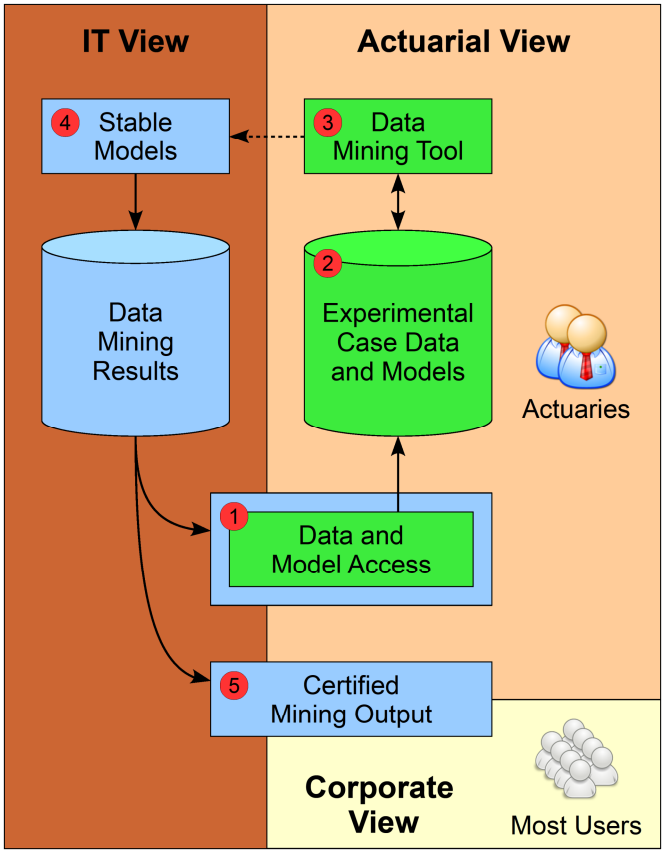

Figure 5 – VLCS data mining: enhanced functionality, on a large scale, managed as a formal corporate asset by IT

The figure reuses the top-level architecture, with the components related to a mining VLCS relabeled to show their purpose, and the components that are unchanged removed for simplicity, though they still apply. Elaborating on the five basic steps of data mining, but stating them in VLCS terms:

1) Actuaries pull data from a number of tables in the DW into the exploration area and join them in SQL into case form to be used for model learning, which is a form of calculation experiment.

2) Some of that data also tests the model. These learning and testing activities likely never become stable calculations since they will always involve creative thought.

3) Actuaries find a model with long-term usefulness to the larger organization, so the deployment step is done as part of an IT release process that includes both the code needed to get the data in case form and the model itself.

4) Each time IT has new DW data, it runs the new data against the model. No parameterization is needed for initial deployment—IT simply runs all the models on a fixed cycle.

5) IT integrates the predicted values into the standard BI tools for both actuaries and the larger user community to consume.

The organization now has a VLCS that codifies its mining work as a permanent corporate asset, but it is only a rudimentary VLCS. The organization can push the VLCS to an even higher level of value by iteratively layering on a variety of higher VLCS functionalities as need and budget allow, including:

1a) Define a variety of case tables that IT can generate as part of the VLCS and allow the selection of which case form to use in the parameter interface.

2a) Have the VLCS test the models as part of its run and store the test scores in separate tables that can be used when judging the output for certification.

3a) Define multiple models and predicted values that mine a wider array of business needs and allow selection in the parameter interface.

3b) Ensure both actuaries and IT use a common tool set so that mining algorithms move from the exploration area to the DW without modification rather than being recoded from the actuaries' language into the IT language.

3c) Expose parts of the model deployment process so actuaries do not have to wait for IT release cycles—that is, parameterize and self-service-enable *the deployment process itself*—a very advanced interpretation of VLCS principles.

4a) Add the ability to specify which mining algorithms to run and the parameters to those algorithms, such as confidence levels, windsorizing factors, decision tree depths, and the hundreds of other parameters that data mining allows.

4b) Add the ability to execute the mining algorithms on-demand after the parameters have been adjusted.

5a) Add status codes so the user community knows what is certified and what is experimental.

4. IT VIEW

Making the actuarial view work efficiently requires critical design and implementation decisions by IT behind the scenes. These decisions are discussed here in a way that makes the direction clear but leaves the details to each particular IT shop since every shop has its own unique tools and processes and the VLCS architecture is agnostic to them.

4.1 Operational Systems

Policy writing systems, claims payment systems, and billing systems are examples of operational systems. A large insurance company will typically have dozens if not hundreds of them. Known more formally as on-line transaction processing (OLTP) systems, such systems generate *operational* data which must be converted into HIGSO form and dimensionalized into *analytical* data to provide maximum analytical efficiency. Such conversion is complex and time-consuming work that a DW is specifically positioned to handle, so in the long-term, the goal should be to move all operational data into the DW and deliver it in dimensionalized HIGSO form. In the short-term, if the need to move a particular operational system's data through the DW is unclear, or a large-scale DW project cannot be funded or will take too long, the data should be placed in the exploration area on an interim basis for R&D purposes. Once established, this exploration use of operational data has the tremendous value of allowing preliminary analysis of the data without the overhead of a major DW project, and of allowing actuaries to learn about the data so they can later specify more precisely how they need it handled by the DW in dimensionalized HIGSO form. Such use must avoid long-term retention of the operational data since this will cause the environment to grow to a size and complexity not easily managed by non-IT areas. As with all data in the exploration area, institutional reporting on this data must be avoided.

Two approaches can be taken to get operational data into the exploration area. The aggressive approach is to point the high-power data access tools at the operational systems and read the data directly into the exploration area. This has the benefit of being easy to set up, but puts a processing burden on the actuaries and challenges operational system access policies, which typically restrict access to well-defined uses by a small number of administrative authorities. A more conservative approach is to employ IT to move the data into the DW and expose it for use largely as-is rather than converting it to dimensionalized HIGSO form. This has the benefit of leveraging the DW's existing tools, processes, and access policies related to acquiring operational data, but does require accepting the time and cost of a small DW project and giving up the tremendous power of

dimensionalized HIGSO form. Such DW work is not throw-away in the long-term (the way the more aggressive approach is), yet it is relatively cost-effective—loading sources into a DW is a small fraction of total DW costs; it is the integration, subject-orientation, and dimensionalization that are expensive. By using the more conservative approach, a balance is struck between moving quickly and positioning for the long-term.

4.2 Data Warehouse

Using a DW to support exploration requires IT to expose the internal business rules of the DW more than usual, using a DW to support a VLCS requires IT to know the proper location for a VLCS in the DW flow.

The exposure of DW business rules is something that the larger user community often wants to avoid. They assume that the normal data warehousing process gathered the appropriate business rules, built them into the systems, is verifying the business rules as part of routine operations, and is certifying the data accordingly—meeting these assumptions is a core DW value, thus a safe assumption. Actuaries engaging in exploration cannot operate on such black-box assumptions. Exploration necessarily looks for new patterns and connections, requiring the current patterns and connections to be understood. How data is grouped, which source "wins" when more than one is providing different data for the same field, how different code sets map to each other, the many variances based on which state of the U.S. is in question—these business-rule-driven manipulations of data in the DW must be published by IT in an easily understood format that is actively maintained.

The proper location for the VLCS in the DW is the location that balances the need for stable data coming in and flexible control going out. There are competing schools of thought about the "data staging area"[8] or "corporate information factory"[9] work that ultimately generates DW output, but for VLCS purposes, the three blocks shown in Figure 6 suffice. "Staging" takes in operational data and transforms it into HIGSO form. "Atomic" holds the HIGSO data, in a manner optimized for system use, but not efficient for BI tool or human use. "Delivery" restructures the HIGSO data into dimensional form so that users and BI tools can efficiently use it for analytics. These areas constitute the basic DW.

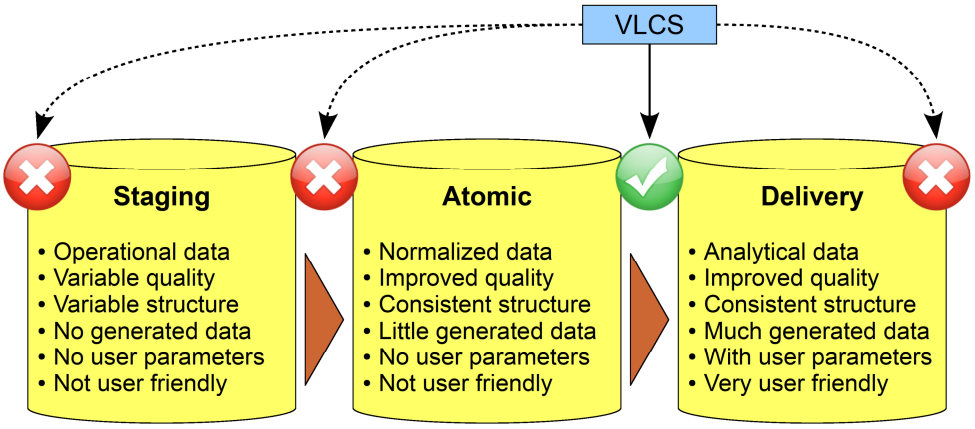

Figure 6 – Data warehouse layers with the proper location for the VLCS

The proper location for the VLCS is after atomic but before delivery. Locating the VLCS before staging is impossible because the data is not yet cleaned up and properly structured, and the staging intake is generally too slow and too controlled for the dynamic, self-service requirements of the VLCS. Locating it before atomic is possible, but this causes the VLCS calculations to take place at the same time as quality and structure improvements, resulting in a level of complexity that is difficult to manage; and atomic work is also still too slow and controlled for VLCS requirements. Locating it after delivery would require doing the calculations against data in dimensional form. While the dimensional form is powerful for BI tools and human thought, it is poorly suited for the computer-processing nature needed for the VLCS. Locating the VLCS between atomic and delivery is ideal—the source data is cleaned up and structured for efficient processing, the parameter loading can be kept dynamic, and the generated data can be efficiently stored and integrated into the dimensional structures. This location is the point of optimal balance between the stable data on the left and the flexibility needed on the right.

4.3 Parameter Interface

The parameter interface allows actuaries to express complex input with enough structure to allow parameter consumption by computer code. There are dozens of solution patterns, but they generally fall into the four categories discussed here.

4.3.1 Parameters by Screens

Data entry screens work well for needs that are well defined and seldom change, such as entering name, address, and other personal information in a Web page. They are rather poor for such things

as rate tables, loss development factors, arbitrary value banding, and other parameter sets that by their nature have lots of categories, ranges, factors, comparative logic, and other points of variance. Proper use of data entry screens for VLCS design tends to occur when the *categories* of parameters are stable but the *values* within those categories need to be arbitrarily selected. For example, if a report is always accessed by coverage, deductibles, and limits, these can each get a component on the interface, but the values in these components would be populated dynamically based either on the data itself, or on reference tables uploaded using one of the more robust techniques below. Screens may also be useful if the formulas are sufficiently simple. In such cases, a basic formula entry screen that looks something like a robust calculator can be written. The major weakness of screens is that they require a comparatively heavy IT engagement, both up front and when things change. The major strength of screens is that they can be constrained to prevent user error, thus allowing the parameterized work to be pushed to junior skill levels.

4.3.2 Parameters by Spreadsheets

Spreadsheets are often the optimal approach for passing parameters and can produce extremely robust parameterization[10]. Formulas and cell referencing provide high expressive power within the spreadsheet itself. The sheet/column/row/cell structure makes spreadsheets unambiguously navigable in computer code. Naming various sections within the spreadsheet makes the computer code human-readable. The ubiquity of code libraries that can process spreadsheets allows IT to choose among many programming languages. That nearly every IT professional and actuary has worked with spreadsheets throughout their career makes spreadsheets a common tool with which everyone is immediately comfortable.

To be done effectively, actuaries and IT must agree on the structure of the spreadsheet, where to upload it, and what to do when it is uploaded. The structure is the most important part. For example, if the computer code expects the names of coverages to be in column C and the limits on the coverages in column D, they had better be there. If the code will assume the first blank cell in a column to be the end of the list, there better be no embedded empty cells, etc. Actuaries must be able to specify a very robust structure for the spreadsheet, and once it is done, they must stick to it. IT should have the VLCS scan the spreadsheet's structure for basic compliance to the agreed format, but at some level, there is a garbage-in/garbage-out (GIGO) risk that is unavoidable and must be addressed through caution by actuaries using the spreadsheet and by IT staff monitoring the processes that consume the spreadsheets for anomalous behavior. Naming of cells and ranges should be used extensively as this greatly enhances human readability and system navigability.

Spreadsheets meeting these requirements are placed on designated locations such as shared drives where an IT process is watching for its arrival. Seeing the parameter file, the process may invoke the appropriate back-end activity including such things as running calculations, reporting run status, and notifying the user when milestones are complete.

4.3.3 Parameters by SQL

Eventually spreadsheets reach a limit because they are not designed for general-purpose data handling, at which point SQL can be used (pre-relational technologies, such as hierarchal databases, are not addressed here, but the concepts are the same). The SQL is written by actuaries based on the tables coming into that point in the processing, and the tables going out. This use of SQL is not interactive the way it is when querying data, but instead is done in advance and uploaded, even if "in advance" is immediately before running the calculations. The interface itself can be in two forms. One is truly free-formed where actuaries can put in literally any SQL needed to achieve their desired outcome. The other is constrained to the major clauses with some filtering to prevent issues. For example, the screen may have an area to list the columns to put in the SELECT clauses, the tables to put in the FROM clause, and the conditions to put in the WHERE clause. The code then puts these together and scans them for undesirable things such as cross joins, non-key joins, and other SQL constructs that are hard on the system and/or logically invalid. As with spreadsheets, there must be agreement about the GIGO risk and the appropriate mitigations should be in place.

4.3.4 Parameters by Code Modules

Anything that can be done in a relational database can be specified and executed in SQL, but some desired outcomes are terribly difficult to do in pure SQL and are much more readily accomplished using a combination of SQL and general purpose programming languages. Such languages come in two major forms from the view of actuaries. The lesser form is macros in office automation tools such as spreadsheets. Since using spreadsheets for parameterization is likely already part of the plan (per the previous section) it is also possible for actuaries to add macros that execute very advanced logic. These macros would be named and then called at run-time by the VLCS. The greater form is major statistical tools such as R, SAS, or SPSS. For these and similar general-purpose tools, actuaries will write the code modules to do whatever logic is needed. Many actuaries likely learned to do such coding in college, and the more adventurous are likely to now do it somewhat routinely in the exploration area. Such code modules, instead of just running on actuaries' local accounts, are enhanced and deployed by IT in a way that makes the code callable by the VLCS. The

enhancements are of a technical nature and do not change the core business logic, which remains the responsibility of the actuaries. Enhancements may include multithreading, error handling and recovery, monitoring and metrics, and other well-defined yet business-agnostic characteristics of IT systems. Callability is facilitated by enterprise integration middleware based on such things as REST, WS-* protocols, and message queuing technologies. Again, GIGO applies.

4.4 Loaded Parameters

The primary responsibility of IT once the parameters have been loaded is to stamp them with user, version, date, and status attributes and hold them for historical reference. Actuaries must be able to go back in time and see what parameters were used and correlate them to the output produced. If the output is eventually promoted into the main data warehouse for use in standard BI tools, the organization must be able to audit the parameters as part of the full lineage of all data under compliance.

4.5 Execution Interface

The execution interface provides self-service invocation of the VLCS on the assumption that actuaries will be rational in the use of the system if properly informed, but as a safeguard, IT must put in back-end components that put constraints on the maximum amount of power actuaries can consume. Database and operating system workload management tools should be used (as discussed shortly), but these generally need to be supplemented because a well-designed VLCS can spawn any number of parallel processes, thus overwhelming any generalized workload management tool. To prevent this, execution should use a customized back-end process that caps and queues submitted jobs and the number of processes per job; for example, a custom-coded UNIX daemon that listens on a TCP/IP port for invocation requests, forking the appropriate process if the cap is not reached, but queuing the request if the cap has already been reached. Such behaviors by the daemon should be table-driven, using tables common to the daemon and the execution interface so that the caps, queue, and current running state are easily read by the execution interface for presentation to the actuaries. Such tables should retain the information indefinitely, providing the historical view also desirable on the execution interface.

4.6 Stable Calculations

Stable calculations are those that are no longer dynamic enough to need the knowledgeable intervention of actuaries and are being done repeatedly enough to match the strengths of IT, thus

making it logical to implement them as formalized business rules in an IT-supported system. Separating these stable calculations from those that still require flexibility is one of the greatest challenges of VLCS design. In the data mining example earlier, the separation was fairly clear—the very advanced mining algorithms are stable across the industry and hidden within the mining tools, but parameters to those algorithms are externalized for the user to adjust. A much harder challenge occurs when the core calculations are to be developed in-house, requiring actuaries and IT to work together to determine what is flexible, what is stable, and how to separate them. The challenge is met by modularizing the steps in the process, clearly quantifying the information going into and out of each step, and outlining the basic logic of each step. An extremely simplified example of the modularized steps and critical information for a loss development process might look like Table 2. The design and build of the VLCS proceeds around this structure.

Modularized Step	Type	Critical Information
Acquire operational data	Stable	Inputs: operational data from around the enterprise. Outputs: operational data in the DW in HIGSO form. Logic: HIGSO processing. Rationale: Normal DW work.
Convert data to actuarial codes	Stable	Inputs: DW HIGSO data. Outputs: HIGSO data with codes that actuarial uses. Logic: Standard data transformation. Rationale: Actuarial needs data coded to their rules E.g. loss bin A = $1-$499, B = $500-$1,249, …
Capture base balancing numbers	Stable	Inputs: DW HIGSO data. Outputs: Aggregated counts and sums. Logic: Counting and summing. Rationale: Capture numbers at an aggregate level now so they can be used for balancing later.
Load loss development factors (LDFs)	Flexible	Inputs: Actuarial spreadsheet with LDFs. Outputs: LDFs in the database, stamped with user, data, and version information. Logic: Basic upload. Rationale: LDFs change based on the knowledge of actuaries and must be under their control.
Load catastrophe identifiers	Flexible	Inputs: Actuarial spreadsheet with date, geographic, and other identifiers of events to be considered catastrophic. Outputs: Identifiers in the database, stamped as above. Logic: Basic upload. Rationale: Actuarial is charged by the organization with classifying catastrophes based on data characteristics.
Develop & store losses in raw form	Stable	Inputs: DW HIGSO data and all uploaded parameters (LDFs and cat. IDs in this example). Outputs: Developed losses in the database. Logic: Calculations based on the standard rules of developing losses. Rationale: Well-defined algorithm for which all inputs have been received in the DW.
Balance against base numbers	Stable	Inputs: Developed losses and base balancing numbers. Outputs: Stratified accuracy percentages. Logic: Basic math. Rationale: Manipulated data often cannot be balanced at a fine grain, but in aggregate, basic "sanity checks" can be made.
Filter categories in/out as needed	Flexible	Inputs: All data from above and a parameter screen. Outputs: Run parameters in the database. Logic: Capture user preference for the run they are requesting. Rationale: Large amounts of loss development data are in the database, but the user needs only a targeted amount. E.g. non-catastrophic losses in Texas for the previous 12 months for auto coverages grouped by coverage limits.
Connect developed losses to BI tools	Stable	Inputs: Developed losses. Outputs: Developed losses presented side-by-side with main DW data in standard BI tools. Logic: Integration logic that matches the developed losses to their original HIGSO subject areas. Rationale: The larger organization may have an interest in actuarial's developed loss work—such as product or regional managers who control what kinds of business to go after.
Generate reports in triangle form	Stable	Inputs: Developed losses and user parameters. Outputs: A triangle-structured report holding just the data the user specified in the parameter screen. Logic: Filtering logic embedded in the SELECT and WHERE clauses used to retrieve the developed loss data; a bit of creative transposition coding to get it to look like a loss triangle. Rationale: Standard format for presenting developed losses.

Table 2 – Modularized steps of loss development and their inputs, outputs, logic, and rationale

4.7 High-Power Data Access & Overall Performance

The *high-power* aspect of data access is determined *far* more from the back-end tooling than by the software the actuaries see on their desktops. As discussed, the only real requirement for high-power data access from the actuarial view is that the tools allow them to run arbitrarily complex SQL. The challenge for IT is to ensure that when such queries get submitted, they run fast. This need for speed and the solutions about to be discussed also apply to some to degree to other contexts, such as making the exploration area and standard BI tools run fast, so this section will address topics that IT should keep in mind whenever they need a fast platform for any user community. The need for speed is satisfied by having sufficient hardware, utilizing it with partitioning and parallelism, connecting it together with high-bandwidth networks, and keeping it all under control with workload management tools.

4.7.1 Hardware Capacity

At some level, computing power comes down to the old racing adage, "speed costs money, how fast do you want to go?" No design, implementation, usage policy or other non-hardware factor will make up for a lack of the raw power that comes from sufficient hardware—but *non-hardware* factors can still produce bad results *despite available hardware*, such as running serially instead of in parallel, not partitioning, hopping repeatedly across the network instead of clustering processing steps on one server, etc. Thus a goal in good VLCS design is to do nothing that limits full and proper utilization of hardware, but sufficient hardware must also exist.

IT must then ensure that the hardware is configured in a manner known as *balanced*. A balanced hardware configuration is one that ensures that all hardware sub-components are operating at the same speed. Disks, disk controllers, motherboards, HBA cards, NIC cards, interconnect switches—these are the small hardware pieces that add up to the total hardware solution. Don't worry about what they all are—only know this: just as a chain is only as strong as its weakest link, the platform hardware is only as fast as its slowest subcomponent. Balancing ensures that all subcomponents are at the same speed. Balancing a large platform is so hard that the IT team within the organization often should not attempt it, instead outsourcing it to the vendor of the platform itself. The vendor is far more likely to be able to balance their platform than any insurance company's internal IT staff. Often such a service is provided for free or at low cost as part of the purchase of the platform, but IT should be sure to write it into the contract specifically.

4.7.2 Partitioning and Parallelism

The *very large* in the VLCS comes in two forms: very large data and very large processing. Very large data requires partitioned data; very large processing requires parallel processing. Partitioned data is data stored on disk in a way that allows the system to retrieve only the data that is used in any particular query. For example, a database may have 10 years of data, but the actuaries are only analyzing the last six months, so only the last six months are retrieved by the database, thus getting a 20x performance boost. Parallel processing is processing done using many processes all at once, each doing a small slice of the same big task. For example, the last six months can be done with six processes each going after just one month, thus getting a 6x performance boost. Overall, the example is 120x faster than one process on all the data. Factors consistently above 100x are not unreasonable on a VLCS built with proper partitioning and parallelism (P&P).

P&P are critical success factors to the VLCS[11]. They are achieved by using tools that can *potentially* handle the very large scale, and then implementing a design that *actually* handles the very large scale. During the tool selection process, vendors must show that every aspect of their suite supports P&P. During system design, actuarial must explain to IT how the data is used (which should already be happening per the same need when separating stability and flexibility). During implementation work, IT must build specific constructs that facilitate P&P. Actuarial must be patient when IT is building P&P since the work will consume time yet not be adding any new business rules, only system performance and scalability.

4.7.3 Networking

Network bandwidth must be high because actuaries will need large result sets or even large bodies of data to be pulled into their exploration area. Bandwidth should be 1-gigabit or 10-gigabit Ethernet to the desktop and 10-gigabit Ethernet or InfiniBand between the DW servers and exploration area servers. Better yet, put the exploration area on the DW server and encourage actuaries to pick tools that run directly on the DW platform, thus eliminating the network hops.

4.7.4 Workload Management

High-power data access, calculation experiments, the execution interface, the freedom of the exploration area, the many queries from standard BI tools being run by a large corporate community, and the repeated admonition to collocate as much as possible for maximum efficiency—these result in a serious risk of overwhelming the platform. Even with sufficient hardware overall, there is still a risk of overwhelming the platform at key times such as right after the

DW publishes the monthly refresh, right before lunch, at the end of the business day, when the boss e-mails a question to the department and a dozen people try to generate the answer independently, and other times when human behavior tends to cluster.

Keeping the platform from getting overwhelmed is the purpose of workload management (WM) tools. As noted, the execution interface benefits from having its own custom-written WM behavior, but in addition to this, WM tools in the operating system (OS) or database management system (DBMS) should also be utilized so that the full collection of processes competing for system resources can be managed holistically. Most OSs and DBMSs have WM tools, and they typically provide the ability to allocate system power based on both statically and dynamically definable characteristics. Static characteristics include such things as username, machine name, or the connection protocol. Dynamic characteristics include such things as how much power a job is consuming, how long a job has been running, or how many jobs are on the system.

IT must implement WM tools as part of platform construction, allocate permanent staff to ensuring WM operates efficiently, and provide easy-to-understand mechanisms for all users of the shared platform to see what is happening on the platform. Actuaries must be realistic about having to share power among the various forms of processing, especially since theirs are the process most likely to consume the most power. As with the execution interface, the hope is that smart users who are properly informed will be realistic about overall organizational priorities and their share of the power.

4.8 Standard BI Tools

The same techniques that allow the DW to connect dimensionalized HIGSO data to standard BI tools also allow generated actuarial data to be connected to standard BI tools. Kimball provides an example[12] so robust that it cannot be improved upon here, so the interested reader is encouraged to review it. In short, the example shows how to connect one of the most advanced forms of calculation work—data mining—to one of the more complex dimensional constructs—aggregated attributes as facts[13]. With such an advanced example established, the feasibility follows for connecting simpler calculations, such as earned premium, to basic dimensional constructs, such as products, geography, and customer.

5. RESULTS AND DISCUSSION

5.1 A Complete and Complimentary Solution Suite

The set of solutions provided here solve any conceivable analytical data problem in a mutually interconnected manner. As a result, there is no analytical data challenge that cannot be handled. *Simple and stable* analytical needs are handled through traditional DW/BI solutions. *Complex and changing* analytical needs are handled through the exploration area. *Complex and stable* analytical needs are handled through VLCS solutions. *Simple and changing* analytical needs are typically done by desktop tools, which were not discussed here since it is a straightforward solution space.

5.2 Reference and Reality

Nearly every aspect of this paper is based on an actual experience of the author—I have worked on data warehouses and exploration areas with dozens of terabytes, calculation engines that could consume every CPU available, interfaces with so many parameters they looked like a DJ's mixing board, rate table spreadsheets with thousands of active cells, process steps that jumped outside the firewall to a third-party vendor and back, an execution interface back-end so advanced it almost became its own little operating system, output that went to the CEO, output that became published rates with state insurance offices, and lots of what-if output that went nowhere—but I have never been on any *single* system that implemented every aspect of the VLCS as detailed here. Likewise, any particular VLCS in any real-world situation will likely not have every aspect. In part this is due to actuarial not needing every aspect and due to time and funding realities when building systems. Rather than being a monolithic goal, the architecture and approach provided here should be used as a broad reference and a large collection of ideas to draw from then narrow down to the needs of a particular situation.

5.3 A Targeted and Challenging Undertaking

The VLCS is a niche solution not to be undertaken lightly. If possible, solve the business problem through some lesser means. Can static reports solve the problem? Can general-purpose slice-and-dice OLAP meet the need? Is there a vendor tool that we can just buy? Can it be done directly from the exploration area, even though audit and legal get nervous about end-user reporting? Odds are, the VLCS will take longer to build and cost more than most people think at the start. Be cautious. That said, do not attempt to meet a VLCS need without a VLCS solution. The VLCS delivers extremely advanced analytical power better than any other pattern. If an organization

is confident that the effort is worth the undertaking, nothing can replace a well-built, highly-customized VLCS.

5.4 Governance and Maintenance Processes

Even the most stable business rules in a VLCS may change from time to time, and new calculations will stabilize in the exploration area and need promoting. These changes should be handled by a small governance team with both actuarial and IT membership. Determining the course of action requires a more open dialog than with traditional IT systems. Traditionally, the business states a need, IT estimates a cost, and the business makes a decision to execute or not. With a VLCS, there must be more of a back-and-forth discussion that explores where the functionality truly belongs. If everyone is comfortable with the stabilization and parameterization that can be provided, move it to IT. If everyone is comfortable with the localized control and do-it-yourself overhead, leave it in actuarial.

5.5 Corporate Standards and the Center of Excellence

The organization has a powerful opportunity for efficiency if it encourages actuarial to use official IT tools well before a promotion discussion occurs. If actuaries use IT servers for their high-power exploration work, IT languages for the code they write, and IT vendor packages for their major solution suites, then moving these over to IT if they prove to be stable will be vastly easier than if IT has to rewrite logic, approve new tools, and staff new skill sets. In return, IT must expand its suite of tools to include those preferred by actuarial and native to the nature of their work. IT must standardize and publish the proper use of those tools, with actuarial being a significant contributor to the standards and IT adding value primarily through formalization and stewardship. IT must staff a small group of technical experts, typically known as a "center of excellence," whose job is to actively support actuaries in the use of tools and standards, as well as perform the monitoring and maintenance of the exploration area previously discussed.

5.6 Build Versus Buy

As noted at the start, the generic nature of vendor tools makes them poor candidates for deepening the competitive advantage of differentiating work—a core effect of the VLCS. But a number of tools in the insurance industry follow the VLCS pattern, whether they call it that or not, and buying a VLCS may make more sense than building one if the packaged solution matches the business need. Verify that the product's overall architecture matches that of the VLCS architecture

in Figure 2, and pay particular attention to the flexibility of the parameter interface, parallelism, partitioning, high-power data access, connecting data to the DW, and integrating data into standard BI tools. If the product seems favorable, give more weight than usual to buying—building a VLCS is hard, and if a vendor has truly done it, buy it from them. If there are any clear faults in the product, give more weight than usual to building—getting locked into a limited VLCS destroys creative knowledge work.

5.7 Higher-End Work with Lower-End Skills

It was noted previously that VLCSs tend to apply to the upper end of actuarial work continuum. Notice, however, that if the calculations are codified into an IT system, the parameter and execution interfaces are sufficiently simple, and the data is hooked into standard BI tools, it is possible to assign more junior actuaries to the use of the VLCS, having them draw in more senior actuaries on an exception basis only. This ability to commoditize advanced work down to a more junior level is one of the most important value-adds of a VLCS because it allows the most senior actuaries to continually drive to the top of the actuarial work continuum. Very often, the core of the business case for a VLCS comes down to the hard dollar cost of building an advanced IT system versus the soft opportunity cost of not using the best actuaries on the hardest problem because they are bogged down in work that should be getting commoditized.

5.8 Agile Software Development

Systems are built using some combination of two basic approaches: waterfall and agile[14]. Waterfall is a sequential process that requires actuarial to create detailed specifications. IT builds to the specifications. Changes along the way are viewed negatively and rigidly controlled. Agile is an iterative process that requires actuarial to specify a small increment of what it needs[1]. IT builds the increment. Both learn from the increment, embrace changes discovered, build the next increment, and repeat until the system is complete. VLCS development should use the most agile approach possible, and it tends to lend itself efficiently to doing so. For example, notice how the data mining example earlier had a basic level then a more advanced level, and how the more advanced level had line items that could be discretely built and delivered to actuarial. It was presented in this form precisely to show how it could be iteratively delivered using agile rather than having to be a massive commitment under waterfall.

[1] Agile is a set of principles. Several methodologies implement these principles. Among the better ones is Scrum[15].

6. CONCLUSION

The VLCS is the most advanced analytical system an organization can build, with a risk/reward profile to match—it will push IT staff to the limits of their skills and IT systems to the limits of their power, and it will require actuaries to have truly unique and useful ideas that the VLCS can help them explore and deliver. But it will also provide a level of value not achievable through more humble means, including optimization of the actuaries' time, maximum reasonable flexibility, legal compliance and auditing, tolerance of staff turnover, the power of a large-scale IT system, deep competitive advantage, and maybe even a little fun—there's just something cool about having direct control over a powerful system running complex calculations at top speed against massive amounts of data!

Acknowledgments

The author is grateful to his many technical peers and business partners who have helped him understand how to engineer a VLCS and utilize it for maximum business value. Special thanks to Doug Anderson who first taught me the pattern, to Ian Penman whose engineering talent showed me the power of a VLCS in its most advanced form, and to Bob Allen who patiently explained the nuances of actuarial logic in terms that even an IT guy like me can understand.

7. REFERENCES

[1] Bukhbinder, George, Michael Krumenaker, and Abraham Phillips, "Insurance Industry Decision Support: Data Marts, OLAP, and Predictive Analytics," *Casualty Actuarial Society Forum*, Winter **2005**.
[2] Inmon, William H., "Building the Data Warehouse," 4th Ed., John Wiley & Sons Inc, **2005**, pp. 29-33.
[3] Kimball, Ralph and Margy Ross, "The Data Warehouse Toolkit: the Complete Guide to Dimensional Modeling," 2nd Ed., John Wiley & Sons Inc, **2002**, pp. 16-24.
[4] Inmon, William H., Robert H. Terdeman, and Claudia Imhoff, "Exploration Warehousing: Turning Business Information into Business Opportunity," John Wiley & Sons Inc, **2000**.
[5] Hewlett Packard, "Rationalizing R&D Sandbox Environments," **2009**.
[6] Brooks, Frederick P., "The Mythical Man-Month," Anniversary Edition, Addison-Wesley, **1995**, pp. 4-6.
[7] Guo, Lijia, "Applying Data Mining Techniques in Property/Casualty Insurance," *Casualty Actuarial Society Forum*, Winter **2003**.
[8] Kimball and Ross, pp 8-10.
[9] Inmon, pp. 402-409.
[10] Krumenaker, Michael, and Jit Bhattacharya, "User Implementation and Revision of Business Rules Without Hard Coding: Macro-Generated SAS® Code," *Proceedings of the 16th Annual Northeast SAS User Group Conference*, **2003**.
[11] Kenealy, Bill, "Insurers Attacking Problems in Parallel," Insurance Networking News, May **2009**.
[12] Kimball, Ralph and Joe Caserta, "The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data," John Wiley & Sons Inc, **2004**, pp. 204-206.
[13] Kimball and Ross, p. 152.
[14] Poppendieck, Mary and Tom Poppendieck, "Lean Software Development: An Agile Toolkit," Addison-Wesley, **2003**, pp 24-25.
[15] Schwaber, Ken, "Agile Project Management with Scrum," Microsoft Press, **2004**.

Definitions

Aggregated	Data that is not granular. For example, if the lowest grain of premium is the earned premium calculated on a daily basis, earned premium at a monthly level is an aggregated form of earned premium.
BI	Business intelligence: processes and tools that enable the business to make informed decisions based on available data.
Bin	A range of data treated as a unit. For example: drivers 18 to 25 years old are in the "Youthful Driver" bin; quartiled information has four bins. Also known as value banding.
Corporate information factory	A conceptual model for how a data warehouse should function. Defined by Bill Inmon, et al. The model advocates treating the intake, processing, and delivery of information in a DW as a factory-like process. It places primary importance on the quality of the data and its structure in the atomic area shown in Figure 6. Competes with data staging area.
CPU	Central processing unit: the computer chip inside every computer that is the primary area of processing. DW servers will have dozens if not hundreds of them.
Cube	An alternative term for a dimensionally modeled data structure. The term "cube" comes from the visual representation showing numerical values in the cells of a cube whose sides are defined by the many dimensions used for drilling around the cells. Visualization beyond three dimensions is difficult. Having more than three dimensions is common for any non-trivial analytical work and is known as a hypercube.
Daemon	A computer process running on a server. Usually runs at all times when the server is running, waiting for requests to come to it through some type of network communication.
Data Mining	The process of discovering previously unknown patterns in large data sets using automated tools guided by users with a reasonable knowledge of the data being mined.
Data staging area	A conceptual model for how a data warehouse prepares its data for delivery. Defined by Ralph Kimball, et al. The model advocates analogizing the intake and processing of information in a DW to a kitchen in a restaurant—it must be clean, efficient, and focused on the real objective—efficient usability in the delivery area shown in Figure 6. Competes with corporate information factory.
Dimensional	A way of structuring, or modeling, data. All data is classified as either a measure or an attribute by which to group measures. For example, "earned premium" is a measure that can be grouped "by state." Attributes are put together in logical groups called dimensions—from which the technique draws its name. For example, all attributes related to vehicles are together in the "vehicle" dimension.
DSS	Decision support systems: systems that provide a concise view of data for business decision makers.
DW	Data warehouse: any large collection of data in HIGSO form. "Large" is relative to the organization and its needs and cannot be precisely quantified generically.
Ethernet	A network communication infrastructure. It is a way of connecting computers so they can communicate. Generally used in 1-gigabit and 10-gigabit speeds. Think of the former as "slow" and the latter as "fast." Competes with InfiniBand.
GIGO	Garbage in/garbage out: An informal term used to summarize a class of problems caused by giving bad input to a system, resulting in bad output. The implication is that the fault lies with the input, not the system. While some garbage-in can be discovered and handled by a system, other input cannot. Garbage input not discoverable by the system is generally the type of input meant when using the GIGO term.
Granular	The decomposition level of data. In general, data should be stored in a data warehouse in the most granular form reasonable. For example, the vehicle dimension should be at the VIN granularity level, not just the year, make, and model granularity level. See also, aggregated.

HIGSO	Historical, integrated, granular, subject-oriented: the primary characteristics of good data warehouse data. Although Bill Inmon provides the elements of this definition, the HIGSO acronym was coined by this paper for simplicity of reference. To prevent acronym-overload, "integrated" may be used in conversation since it is the dominate consideration of the four characteristics.
InfiniBand	A network communication infrastructure. It is a way of connecting computers so they can communicate. Competes with Ethernet. Think of it as "very fast" when comparing to Ethernet.
Integrated	Data that is composed from more than one source, converging the sources on the same definition. For example: a Web site captures marital status as M, S, D, W, O. A call center system captures it as 0, 1, 2, 3, 4. The warehouse integrates and transforms these into Married, Single, Divorced, Widowed, Other.
IT	Information technology: components such as computers, systems, and technologies as used in business contexts; name of the department whose primary responsibility is the implementation and maintenance of such components.
Join	A way of connecting data with other data. It can be thought of as a side-by-side connection, such as joining driver, vehicle, and loss information to see how much youthful drivers in red cars are costing the company.
Message queuing	A design pattern that allows computers to talk to each other. One computer will send messages to another, and the messages will queue-up until the receiving computer can get to them. By loose analogy: an e-mail inbox is a message queue system for human beings.
MIS	Management information system: a system that provides a very simplified view of information to more senior-level managers.
OLAP	On-line analytical processing: systems that facilitate data analysis on large amounts of data with fast response time.
OLTP	On-line transaction processing: operational systems that bring in data one transaction at a time
OS	Operating system: The software that allows computer hardware to function. Microsoft Windows is the operating system familiar to most people.
Pivot table	A simplified dimensional presentation of data. A pivot table provides a very useful subset of the operations associated with dimensionally modeled data. It is generally not as full-featured as a standard BI tool, but it is easier to create, for example in Microsoft Excel with just a few minutes of effort.
Predictive Analytics	A form of analysis stronger than the usual drilling, slicing, dicing, trending, and statistical analysis done on dimensional data, but weaker than data mining; predictive analytics falls somewhere in the middle. The dimensionally presented HIGSO data is utilized, but the user is given the opportunity to apply basic data mining with such restrictions as using only mining algorithms that do not require specialized data preparation, having only default parameters used, applying only one mining algorithm rather than a chain of them, etc.
REST	Representational state transfer: a mechanism of computer-to-computer communication both advocated and criticized for its simplicity. Competes with WS-*.
Server	A computer that does work for other computers. Typically, a high-power computer in an IT data center designed to do work other than that directly related to user interaction. Often runs the UNIX operating system, though Windows can also run on servers.
SLA	Service level agreement: an agreement between IT and a business area that quantifies the operational characteristics of a system, such as how often it runs and how long it takes to complete when it does.

SQL	Structured query language: the language used to access data on nearly all modern database systems in a business environment. Older systems still exist that use non-SQL data access languages, but these will continue to fade over time.
Star schema	An alternative term for a dimensionally modeled data structure. The term "star" comes from the visual representation that comes from showing a table of numerical values in the middle and the many dimensions used for drilling around the outside like points in a star.
Subject-oriented	Data configured to the entities, concepts, and language understood by the business when doing analysis. Contrasts most sharply with the source-oriented form in which most operational data is received by the data warehouse.
TCP/IP	A network communication protocol. It allows computers to communicate with each other in a very consistent manner. It is the protocol that underlies the Internet as well as most internal corporate networks. Runs on top of Ethernet and InfiniBand.
Tool	A deliberately loose term used to encompass any combination of software, hardware, vendor applications, custom code, queries, data, algorithms, processes, and other solution elements gathered together to solve a problem. "Tool" is used heavily in this paper because complex solutions often require a diverse combination of specific component types; to state any one is often inaccurate; to keep listing all the components is cumbersome. Classic example: the performance a user sees in "standard BI tools" rarely has to do with the BI software (i.e., visible software); it is actually due primarily to the ability of the database (i.e., more software) to stripe the data across many drive arrays (i.e., hardware) according to the partitioning setup by IT (i.e., design), as well as the interconnects (i.e., networking) between database server nodes (i.e., more hardware) that facilitate data movement; thus it is better to say "BI tools" unless "BI software" is specifically what is meant.
UNIX	An operating system common on servers in IT data centers. Linux is generally regarded as a form of UNIX, though purists might argue otherwise.
VLCS	Very-large calculation system: the term coined for an IT system that fits the design pattern detailed in this paper. To prevent acronym-overload, "big calculator" may be used in conversation since this term captures the essence of what the VLCS is.
Windsorize	An outlier handling technique where outliers are placed into the bin closest to their extreme. E.g., data points above the 95th percentile are placed in the 95th percentile bucket and data points below the 5th percentile are placed in the 5th.
WS-*	Web service standards: a collection of dozens of standards produced by the World Wide Web consortium (W3C) for communication among computers. Competes with REST.

Biography of the Author

James Madison is a senior information architect at a large insurance company. He has managed, designed, and built various VLCS systems during more than a decade of insurance industry work. He is an adjunct professor of computer science at Rensselaer Polytechnic Institute. He can be contacted at madjim (at) bigfoot.com.