

# The Case for Business Modeling



**Hired Brains, Inc.**

By Neil Raden  
February 2005

## ABOUT THE AUTHOR

Neil Raden is the founder of Hired Brains, Inc./Hired Brains Research. Hired Brains <http://www.hiredbrains.com> provides consulting, systems integration and implementation services in Business Intelligence, Data Warehousing, Performance Management, Advanced Analytics and Information Integration for clients worldwide. Hired Brains Research provides consulting, market research and advisory services to the technology industry. Based in Santa Barbara, CA, Neil Raden is an active consultant and widely published author and speaker. He welcomes your comments at [nraden@hiredbrains.com](mailto:nraden@hiredbrains.com).

Neil Raden began his career as a property and casualty actuary and has spent his entire career involved with modeling and analytics. He wishes to thank Applix for this opportunity to express his thoughts on a subject he is passionate about.

© Hired Brains, 2005. All rights reserved.

No part of this document may be reproduced in any form, by Photostat, microfilm, xerography, or any other means or incorporated into any information retrieval system, electronic or mechanical, without the written permission of the copyright owner. Inquiries regarding permission or use of material contained in this document should be addressed to:

Hired Brains, Inc.  
2777 Exeter Place  
Santa Barbara, CA 93105

# Table of Contents

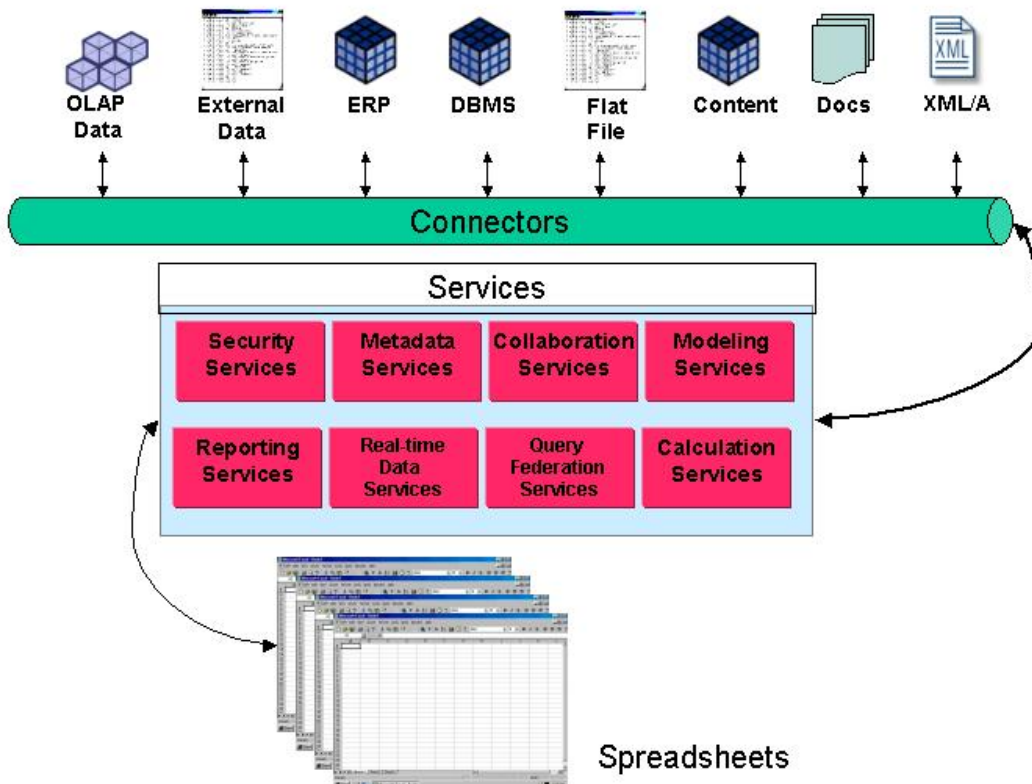
Executive Summary .....	1
The Need for Business Models .....	2
BI Falls Short .....	2
BI and Modeling .....	3
Business Modeling .....	3
Modeling Requirements .....	4
A Hierarchy of Analytical Models .....	6
Multidimensional Models .....	6
OLAP .....	7
Spreadsheet Models .....	7
Non-procedural Models .....	7
A Complex Model Example .....	8
The Model .....	10
The Data .....	11
Modeling .....	11
Metrics .....	12
Conclusion .....	14

## Executive Summary

While data warehousing and Business Intelligence (BI) have propagated over the past decade, surveys reveal that the reach of these technologies inside organizations is largely superficial. At the same time, spreadsheet use continues to grow. One hundred and fifty million users of Microsoft Excel are evidence that there is an unmet need for the BI industry to provide more business modeling capabilities. Though modeling was front and center in the age of Decision Support Systems 15-25 years ago, today BI is more concerned with reporting, Performance Management, digital dashboards and, for a small number of power users, data surfing with OLAP tools.

The need for business modeling is greater than ever. Never before have organizations been faced with the magnitude of challenges such as visible supply chains, connected customers, internationalization and ever-greater regulatory issues, to name a few. The deployment of personal productivity tools that lack enterprise capabilities to do the job, such as spreadsheets and personal databases, is a major problem with colorful names such as spreadsheet hell, Shadow IT and spreadmarts.

Spreadsheets are endowed with a magical quality, though, that is vital to modeling – their expressiveness. Business users gravitate to spreadsheets because they're inexpensive (at first), ubiquitous, autonomous and convenient, but their greatest quality is the way anyone can approach a model and render it in a spreadsheet easily. That is their gift. If one were to design the perfect solution to business modeling, it would preserve all of the desirable qualities of spreadsheets and relocate all of the other aspects to a more controllable environment. A vastly simplified architecture might look like the schematic below. In this paper, the requirements of effective business modeling are described and the case for better modeling tools is made.



## The Need for Business Models

In the past fifteen years, great strides were made to provide more data to more people more easily. To a large extent, these efforts were successful. However, there was an almost single-minded fixation on data, databases, data warehouses, data marts and data management in general. As a result, so many resources were absorbed that the tools and methods to leverage that data did not progress at the same rate. In fact, many of the so-called Business Intelligence (BI) tools provide so little functionality, from a business user's perspective, that one could argue that this fifteen-year effort has only set the stage for the next act waiting patiently in the wings. Evidence of this can be found in the continually expanding role of spreadsheets used for most modeling and reporting tasks, even in those organizations with the most sophisticated and well-known data warehouses.

Acceptance of new technology, in organizations and in an economy as a whole, is never smooth. It is not uncommon for one set of technology innovations to clearly imply another, seemingly obvious follow-on, yet the acceptance of the next step lags and languishes. There is evidence of that now in data management. Advances in the capacity and relative cost of computer and storage hardware have allowed more data to be handled than ever before. In ten years, it's grown three or even four orders of magnitude. Today's ubiquitous network and nearly limitless amounts of reliable bandwidth permit more data to be managed both centrally and in a distributed manner. Standards such as XML, web services and SOAP will simplify the cooperation of different applications and pave the way for entirely new types of applications.

However, innovations for the people who use data have not kept pace. The process of finding, understanding and manipulating data is still time consuming and tedious for many. While the underpinning of Business Intelligence, information management, is vibrant and dynamic, the user-facing part of it still needs a healthy dose of innovation.

### ***BI Falls Short***

Recently published research by Hired Brains Research<sup>1</sup> indicates that user-facing BI software (analytics for short) is deficient in two major categories: relevancy and understanding. In follow-up interviews, Hired Brains was able to determine the root causes of these failings:

- Analytics tools were too closely aligned to the physical representation of data in the larger data stores, typically relational databases, creating a barrier to understanding from two perspectives: the technical (columns, tables, joins, views, etc.) and the semantic (what does this data really mean and what is its relationship to other data?).

---

<sup>1</sup> *Intelligent Enterprise Magazine*, "Dashboarding Ourselves," November, 2003.

- An inability to manipulate the data and create models with it as most BI tools are read only and lack true modeling capabilities
- A lack of cohesion between the analytics and other work already in place; an inability to close-the-loop
- Stagnation of the (data warehouse) data model once implemented, hindering the ability of business people to continuously refine and update their value-added models or, at the very least, creating an inefficient and expensive-to-maintain process of workarounds.

## ***BI and Modeling***

In summary, business people need more than access to data: they require a seamless process to interact with it and drive their own models and processes. In today's environment, these steps are typically disconnected and therefore expensive and slow to maintain, independent of data quality controls from the IT side and, unfortunately, absolutely indispensable. The solution to the problem is a better approach to business modeling coupled with an effective architecture that separates physical data models from semantic ones. In other words, tools business people need to address physical data through an abstraction layer that allows them to concern themselves only with the meaning of data, not its structure, location or format.

BI isn't standalone anymore. In order to close the loop, it has to be implemented in an architecture that is standards-based and it has to be extensible and resilient, because the boundaries of BI are more fuzzy and porous than other software. While the preponderance of BI applications in place is fairly static, the ones that generate the most value for companies adapt quickly. In the past, it was acceptable for individual "power users" to build BI applications for themselves or their group without regard to any other technology or architecture in the organization. Over time, these tools became brittle and difficult to maintain because the initial design was not robust enough to last. Because change is a constant, BI tools need to provide the same interactivity at the definitional level that they currently do at the informational level. The answer is to provide tools that allow business people to model.

## **Business Modeling**

The term "business modeling" usually conjures up the image of MBA's with a quantitative bent developing indecipherable programs that only they can understand. In point of fact, all business people use models, though most of them are *tacit*, that is, they are implied, not described explicitly. Evidence of these models can be found in the way workers go about their jobs (tacit models) or how they have authored models in a spreadsheet.<sup>2</sup> The problem with tacit models is that they can't be communicated or shared easily. The problem with making them explicit is that it just isn't convenient enough yet. Most business people can conceptualize models. Any person with an

---

<sup>2</sup> A spreadsheet model is very definitely explicit, but it has built into it many assumptions that are tacit.

incentive compensation plan can explain a very complicated model. But most people will not make the effort to learn how to build a model if the technology is not accessible.

There are certain business models that almost every business employs in one form or another. Pricing is a good example and in a much more sophisticated form, yield management, such as the way airlines price seats. Most organizations look at risk and contingencies, a hope-for-the-best-prepare-for-the-worst exercise. Allocation of capital spending or, in general, allocation of any scarce resource is a form of trade-off analysis that has to be modeled. Decisions about partnering and alliances and even merger or acquisition analysis are also common modeling problems. These are all types of business models.

Models are also characterized by their structure. Simple models are built from input data and arithmetic. More complicated models use formulas and even multi-pass calculations, such as allocations. The formulas themselves can be statistical functions and perform projections or smoothing of data. Beyond this, probabilistic modeling is used to model uncertainty, such as calculating reserves for claims or bad loans. All of these models are still just variations of the first type. When logic is introduced to a model, it becomes procedural and mixing calculations and logic yields a very potent approach. The downside is, procedural models are difficult to develop with most tools today and are even more difficult to maintain and modify because they require the business modeler to interact with the system at a coding or scripting level, something for which most business people lack either the temperament or training or both. It is not reasonable to assume that business people, even the “power users,” will employ good software engineering technique, nor should they be expected to. Instead, the onus is on the vendors of BI software to provide robust tools that facilitate good design technique through wizards, robots and agents.

### ***Modeling Requirements***

For any kind of modeling tool to be useful to business people, supportable by the IT organization and durable enough over time to be economically justifiable, it must either provide directly or interact seamlessly with the following capabilities:

- **Level of expressiveness** sufficient for the specification, assembly and modification of common and complex business models without code; the ability to accommodate all but the most esoteric kinds of modeling
- **Declarative method**, such that each “statement” is incorporated into the model without regard to its order, sequence or dependencies. The software, freeing the modelers to design whatever they can conceive, handles issues of optimization.
- **Model visibility** to enable the inspection, operation and communication of models without extra effort or resources. Models are collaborative and unless they can be published and understood, no collaboration is possible.

- **Abstraction from data sources** to allow models to be made and shared in language and terms that are not connected to the physical characteristics of data and further, to allow the managers of the physical data much greater freedom to pursue and implement optimization and performance efforts
- **Horizontal and vertical extensibility** because many modeling tools today are too closely aligned with either vertical applications (CRM for example) or horizontally in a market niche, such as desktop OLAP or dashboards. *Extensibility* means that the native capabilities of modeling tool are robust enough extend to virtually any business vertical, industry or function, or in any analytical architecture.
- **Closed-loop processing** is essential because business modeling is not an end-game exercise, or at least it shouldn't be, it is part of a continuous execute-track-measure-analyze-refine-execute loop. A modeling tool must be able to operate cooperatively in distributed environment, consuming and providing information and services in a standards-based protocol. The closed-loop aspect may be punctuated by steps managed by people, or it may operate as an unattended agent, or both.
- **Continuous enhancement** is a requirement borne of two factors. First, with the emerging standards of service-oriented architectures, web services and XML, the often talked-about phenomenon of organizations linked in a continuous value chain with suppliers and customers will become a reality very soon and will put a great of pressure on organizations to be more nimble than ever. Second, it has finally crept in to the collective consciousness that development projects involving computers are consistently under-budgeted for maintenance and enhancement. The mindset of "phases" or "releases" is already beginning to fray and forward-looking organizations are beginning to differentiate tool vendors by their ability to enable enhancement with extensive development and test phases.
- **Zero code:** In addition to the fact that most business people are not capable of and/or interested in writing code, there exists sufficient computing power and reasonable cost to allow for more and more sophisticated layers of abstraction between modelers and computers. Code implies labor, error and maintenance. Abstraction and declarative modeling implies flexibility and sustainability. Most software "bugs" are *iatrogenic*, that is, they are introduced by the programming process itself. When code is generated by another program, the range of "bugs" is reduced to the domain of interaction, that is, the dialogue between modeler and computer.
- **Core semantic information model (ontology):** Abstraction between data and people and programs that access data isn't very useful unless the meaning of the data and its relationships to everything else are available in a repository.

- **Collaboration and workflow** is the key to connecting analytics to every other process within and beyond the enterprise. A complete set of collaboration and workflow capabilities supplied natively with a BI tool is not necessary, though. Rather, the ability to integrate (this does not mean “be integrated,” which implies lots of time and money) with collaboration and workflow services across the network, without latency or conversion problems, is not only sufficient, but also preferred.
- **Policy** may be the most difficult requirement of them all. Developing software to model policy is tricky. Simple calculations through statistical and probabilistic functions have been around for over three decades. Logic models that can make decisions and branch are more difficult to develop, but still not beyond the reach of today’s tools. But a software tool that allows business people to develop models in a declarative way to actually implement policies is on a different plane. Today’s rules engines are barely capable enough and they require expert programmers to set them up. Policy in modeling tools is in the future, but it will depend on all of the above requirements.

The popularity of spreadsheets today, with all of their attendant and well-publicized drawbacks as enterprise decision support tools, rests solidly on their expressiveness and declarative nature. A declarative modeling tool means, simply, the gap between the way the modeler expresses herself and the syntax in which the tool represents the model is small. In other words, to build a spreadsheet model that compares cash flows over 10 years using three different methods, simply open a spreadsheet and in a few clicks and a little typing, build it, examine it, print it and email it to someone else. In the next section, we’ll describe a much more complicated model and explain how it would be built with a declarative modeling tool.

## **A Hierarchy of Analytical Models**

Any attempt to categorize analytical models into neat buckets will fail to account for the nuance and overlap between them, but it is useful nonetheless to have a vocabulary for discussing the different types of models and how they are used. It may seem counter-intuitive to place multidimensional models at the bottom of the hierarchy and to place spreadsheet models above OLAP, but the reasons are based on the internal complexity of the modeling concepts, not the complexity or robustness of the software program.

### ***Multidimensional Models***

Multidimensional models group numerical variables by tabs or dimensions. For example, we may wish to examine sales in certain time increments and allow for the aggregation of the data through hierarchies in the dimensions, such as days to month, or product to division. These models are usually very simple and require only defining the hierarchies and some easy derived variables, using arithmetic on other numeric variables. Complexity arises when non-additive values are involved, such as headcount or account balances, which are not additive over time, or ratios and percents, which must be recast in subtotals. Most tools in this category can handle these complications

gracefully. Multidimensional models, as distinct from the next category OLAP, are strictly read-only and used for reporting.

## **OLAP**

OLAP is multidimensional modeling in motion, providing a means for people to interact with the model and manipulate it in real-time. Beyond that distinction, there is a wide range of features and architectures for OLAP that differentiate one product from another. For instance, some allow for interactive update of data, also known as “write-back.” Some provide their own multidimensional database (MOLAP) while others launch queries against relational databases (ROLAP). Some OLAP tools employ a model no more complicated than simple multidimensional while others allow for application-specific features, such as currency conversion and financial accounting functions (allocations and eliminations). The complexity and power of filtering the data also varies widely:

**Simple filter:** where product is “shirt”

**Complex filtering:** where product is “shirt sold more than 90 days past an in-store promotion of shirts in stores open more than one year and shirts were in stock”

The types of queries can also vary widely, such as:

1. Print monthly P&L for division 2
2. Show projected P&L for division 2 based on promotion schedule 3

Multidimensional models and OLAP both lack some features that are present in the lowly spreadsheet though.

## ***Spreadsheet Models***

Spreadsheets are almost infinitely expressive. Models can be built on the fly or, one could say, created without modeling at all. Multidimensional and OLAP tools are highly symmetrical which means they define a basic model and replicate it across dimensions. OLAP tools are not adept at exceptions, missing pieces or ad hoc arrangements of data and calculations. Macros can be built without writing code, allowing the spreadsheet to more or less mimic what an analyst might do manually. Of course, macros can be exceedingly complicated and verbose, and each new release includes more powerful features. But the problem is that data, models, formatting and workflow are all bound up together and it is difficult to isolate the models for publication, comment or collaboration. With all of their expressive qualities, one could say that, as for as the models go, spreadsheets are read-only. A major benefit of spreadsheets is the range of modeling possible by people with only limited training, but once models are entered, it is devilishly difficult to extract them, hence the label “read-only.” Data, of course, can always be entered.

## ***Non-procedural Models***

The most difficult models for software programs to manage are predictably the easiest models for people to express. These are the so-called *non-procedural models*, named

because they can be built declaratively in any order. Powerful non-procedural modeling languages were available and widely used over 25 years ago, such as Express and IFPS, for example. They were host-based tools, lacking the scale and GUI's of today's tools, but the facility with which business analysts could build very sophisticated non-procedural models surpassed most BI tools available today.

Consider the simple Profit & Loss Statement below. When closing the books on a period, the values for each account are known (or are summed from lower-level detail) and the computed values can be calculated easily. However, what happens when there is a need to project a set of financials into the future? Are the calculations the same? Suppose borrowings are dependent on net income. Net Income cannot be calculated until interest expense is known, and interest expense cannot be calculated until net income is known. In addition, a Profit & Loss Statement should only cover a discrete period, but when it represents a forecast, opening balance and closing balance, issues are very much at play. The engine must solve the model iteratively, in this case, using simultaneous equations. Because these relationships are easy to state in business terms, a useful modeling tool should use a declarative interface, hiding the complexity of the actual processing from the modeler. Without a declarative modeling interface, setting up these models requires tedious scripting, a deterrent to propagation of BI and a multiplier of the cost to deploy applications.

	<b>Curent Year</b>	<b>Projected Year</b>
Sales revenue	\$ 8,976	\$ 11,200
Cost of goods sold	<u>5,296</u>	<u>6,400</u>
Gross profit	<b>3,680</b>	<b>4,800</b>
Operating expenses	2,441	3,000
Selling G&A expenses	<u>637</u>	<u>1,100</u>
Operating profit	<b>602</b>	<b>2,700</b>
Interest expense	<u>201</u>	<u>164</u>
Income before taxes	<b>401</b>	<b>2,536</b>
Income taxes	<u>160</u>	<u>1,200</u>
Net income	\$ <b>241</b>	\$ <b>1,336</b>

- ▲ - Calculated values
- ▲ - Solved simultaneously

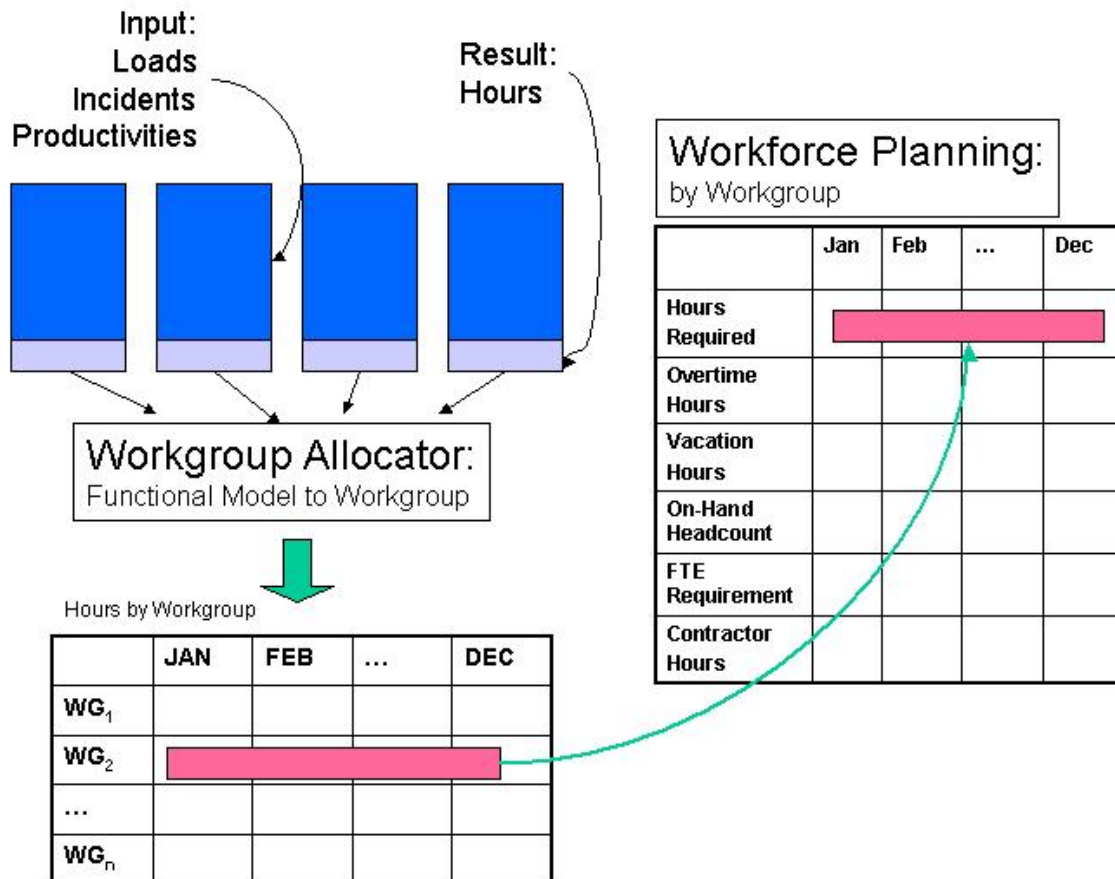
### A Complex Model Example

The following is a real example of a very useful business model and examines how it could be built, maintained and enhanced using declarative modeling. The problem is a common one, of budgeting and allocating human resources for an uncertain set of

conditions, and minimizing the use of contractors and overtime, which are pure added cost. In this case, the problem is the repair and maintenance of a telephone company's physical assets, especially overhead wires and telephone poles, the two most costly aspects of the network to maintain. The problem is complicated by the fact most of the crews are unionized and have very specific work rules. There is a fair amount of "trade-off" analysis that has to occur, such as, "If I shift people from this work group to another work group, will I be covered within a margin of error for the summer tornado season?"

This is not a simple model. Prior to developing the models in a computer, the people responsible for planning and tracking the maintenance and repair budget had to struggle with pencil and paper and non-conformed spreadsheet models, all based on rules of thumb and disparate tacit knowledge.

There are two separate models in the application. The first builds and tracks forecasts of hours of labor needed based on models of the different maintenance, repair and construction requirements. Hours are decomposed in work groups (and aggregated over functions) in order to match hours with actual workgroups. Changing assumptions, doing what-if analysis and working out the optimizations can take place in either model and changes are reflected throughout all of the models. The graphic below shows the basic structure of the models.



Solving the problem involves four steps:

1. Calculating the total amount of labor needed, by time period (monthly for example) and location for a projection period (a year or more) by function, that is, what has to be done: fix a downed wire, clear a lost signal, clean a switch (there are still mechanical switches). Needless to say, these functions are modeled by time, as we mentioned, and by location.
2. The functional models are created at an aggregated level because different “work groups” composed of different people, practices and even unions can be involved in a particular function. For example, if cable buried underground in concrete ductwork has to be repaired, different crews will do the excavation and the cable work. The hours generated have to be decomposed and transposed into work groups, the actual pools of people and equipment in place to address these problems
3. Once the hours are mapped to work groups, additional labor-specific variables are added, such as contractor triggers, overtime targets, utilization percentages, union-specific work practices and, obviously, headcounts with loaded costs of benefits, etc.
4. Actual data can be mapped from source systems to both the first and third steps for both seeding of the models as well as tracking of the results.

### ***The Model***

For the sake of brevity, we’ll only look at the first step to illustrate how to solve this problem with a declarative modeling tool. The models of functions (functional models) are built by the analysts, without code, using a canonical model something like:

```
Load 1:  $L_1$   
Load 2:  $L_2$  (Optional)  
Troubles:  $T$   
Productivity 1:  $P_1$   
Productivity 2:  $P_2$  (optional)  
Hours: =  $(L_1 * T * P_1) + (L_2 * T * P_2)$ 
```

More complicated functions may have more than one Load/Productivity pair. For the sake of simplicity, let’s look at a single load model:

Load 1: Miles of overhead wire (1000)

Troubles: 0.014 (0.14 troubles per miles of wire per month)

Productivity: 11.5 (11.5 hours of labor to clear the problem)

Hours: 161 hours per month needed

## The Data

The grain of the functional model data is hours by month, by function (maintenance, repair and construction) and a few other dimensions, as shown in the table below. In the next step, the data is allocated to actual work groups, by formula, to create a table of hours, by workgroup, by month. In the final step, each row of this table becomes the first row of another model, where required hours are lined up with real headcount and policies for overtime, contractors and even union work rules. In addition, the firing order of the rules has to be reversed when actual data is imported into the model. (Productivities and troubles would be derived from actual loads and hours.)

	Funcnl Model	Function	Geo	Period	Scenario	WorkGroup
Load1	x	x	x	x	x	
Load2	x	x	x	x	x	
Troubles	x	x	x	x	x	
Prod'ty1	x	x	x	x	x	
Prod'ty2	x	x	x	x	x	
Funcnl Hrs	x	x	x	x	x	
WG Alloc%		x	x		x	x
WG Hrs			x	x	x	x
Hours			x	x	x	x
Overtime%			x	x	x	x
Overtime Hrs			x	x	x	x
Contractor %			x	x	x	x
Contractor Hrs			x	x	x	x
Avg Hrly \$			x	x	x	x
Avg Ctr \$			x	x	x	x
Wages			x	x	x	x
Ctr \$			x	x	x	x
etc.			x	x	x	x

## Modeling

Because the functional models are in canonical form, analysts can create or delete instances of the models as they see fit. At this stage, a load, trouble or productivity could mean anything, and in a declarative fashion, the model can be specified as follows:

1. Create a new functional model named "Lightning Strikes of Overhead Wire"
2. System assigns unique identifier (URI) for the model
3. "Load 1" is named "Miles of overhead wire" and the data can be located by indirection through a URI in the semantic information model

4. "Troubles" is named "per mile of wire"
5. "Productivity" is named "Clearing time"

How this would be implemented, as a dialog, a wizard or form, or some other method, doesn't really matter. What is important is that the model was created by simply understanding the concepts, not the technology. Nowhere in this process does the business person use or need to know things like connection strings, instance names, attributes or indexes. Because the interface between the declarative modeling environment and the actual data resources is handled by the abstraction layer, data administrators and managers have complete freedom to relocate, distribute, reorganize and/or centralize the data any way they see fit, provided the process is always synced with the semantic model. The business people receive all of the benefits of optimization, security, data quality and low latency without needing to be involved in the effort. The interface could be a browser or a spreadsheet or even voice recognition (perhaps someday). The engines for calculation, data management and presentation should be neatly separated from this interactive interface.

Consider how this would be accomplished with most of today's BI tools. First, there would be batch processing, running bulk calculations with lots of latency and downtime. Rolling up is one problem: it's a hierarchical dependency. But allocating and transposing is more complicated as the dependencies run vertically and horizontally. It's a highly sensitive model where everything that happens has an effect on almost everything else. Sensitivity analysis and what-if can be run from the beginning or the end or both. To solve this problem today requires lots of power, speed and calculation smarts. To be truly effective, dependencies in the model should be recognized by the modeling tool itself, and changes in one part of the model should flow through and update every dependency without delay.

To keep all this in sync and working in a useful, closed-loop way requires:

- A real-time calculation engine
- A robust, innate understanding of dimensional models
- A calculation functionality smart enough to sift through the declarations and understand what to calculate in what order
- A real-time pump to and from operational systems such as SAP, e.g.
- A standards-based architecture that can cooperate in the closed loop.

## Metrics

An emphasis on compliance and transparency in recent years, due to regulations like Sarbanes-Oxley, HIPAA and Basel II, has elevated discussions about metrics, KPI's and digital dashboards to top-of-mind in most organizations. In fact, these elements have become so important they have a name, Business Performance Management (BPM). The whole concept behind BPM is that careful monitoring of metrics can and will

lead to sustainable improvement. Key to this success, however, is making sure that you watch the right metrics.

How do you know if metrics are correct, or useful or prescriptive? The most important thing to keep in mind is that metrics aren't reality; they are just proxies. We can never measure precisely the things we really want to know, and some metrics are better than others. But there is always the danger that managing to the metric can lead to distortion and dysfunction<sup>3</sup>. Unless a metric is an almost perfect representation of the underlying phenomena, chasing it can lead you in the wrong direction. Also human beings, unlike machine tools or jet engines, know when they're being measured and are ingenious at meeting the metric but not changing their behavior in the desired way. This is manifested in simple ways like sandbagging sales in a good quarter or booking revenue improperly, to more complex dysfunctional behaviors.

Here is a classic example: a commercial bank decided to implement a CRM system. Though the outward message was improved customer service and relationship building, the true motivation was straight cost reduction. Customers hated it and began to desert the bank. They could no longer call their branch; the direct numbers were unplugged and could only be reached through an annoyingly long IVR menu. Because the most credit-worthy customers were the first to go (as they were most susceptible to the offers of competitor banks), the loan portfolio began to deteriorate. Reserves for bad loans are set on a formula (average) basis, so as the loans left, reserves are "released" straight to the bottom line in proportion to the total portfolio, not its relative quality. (With 10% of the loans leaving, the bank should have released < 10% of the reserves.) This temporarily inflates profits while drifting into an under-reserved state, and, of course, profitability was the metric used for executive incentive compensation. The executives collected fat incentives for a couple of years until the portfolio deteriorated so badly that the reserves had to be bolstered, wiping out the previous two years' profits. Ironically, because profitability was going up, they mistakenly believed their CRM effort was a big success, despite what common sense should have told them.

Which leads to problem #2 with metrics: it isn't the metric itself that is valuable; it's how it lines up with other metrics. The trouble with most dashboards is that metrics are usually presented as freestanding, atomic units and their relationships are rarely considered. Devising useful metrics is a critical step, but it has to be preceded by:

1. Finding out what problems need to be solved, which implies...
2. The models needed to evaluate them, which in turn implies...
3. What metrics are needed for the models

This refers to business models, not data models, which are unfortunately the usual first step in most organizations due to the overemphasis of data management, data warehousing and data provisioning. Data models to support analytics must be secondary to real business models.

---

<sup>3</sup> Hope and Hope, "Transforming the Bottom Line: Managing Performance with Real Numbers," 1996, Harvard Business School Press.

One other thing to consider is that metrics are linear and almost everything that has to be monitored is not linear — it's hopelessly complex. Simplifying the problem has to happen because our brains can deal with some complexity, but we haven't figured out how to communicate it to other people. How you simplify it is worthy of some very careful thinking. Metrics are only useful if they can be presented in a context of models, so that they can be explored and investigated as a set interlocking things, not a collection of facts. They are all derived from a dynamic system, with dependencies, causes and effects, many with leading and lagging effects across time. Most BI tools available today have no inherent ability to assist in building coherent models. For example, a planning system that allows separate units to input projections might not have tradeoff analysis built in, allowing, say, three divisions to all ramp up production at the same time without exhausting scarce resources, such as capital or perhaps corporate engineering.

## **Conclusion**

Extracting and cleansing mountains of data may still be hard work, but it is largely a solved problem. Existing approaches to “provisioning” data to users are short-sighted and of limited value. Despite all of the investment in data warehouses, now even enhanced with data federation, for example, most people in organizations still do most of their modeling in their heads because there is no alternative. While spreadsheets are the most capable modeling tool at present, the drawbacks to deploying them as anything other than personal productivity tools are well known, described with such colorful terms as spreadsheet hell, Shadow IT and spreadmarts.

Applying spreadsheets constructively, promoting their use as the visual and navigational interface to true modeling services is an approach with promise. The rise of standards-based service-oriented architectures in the next few years increases the likelihood that robust business-oriented declarative modeling tools will proliferate. Some forward thinking companies like Applix have already begun to implement this approach to positive reviews from the analyst community as well as their customers. They should be commended.