

white paper

semantic discord and data (dis)integration

Michael R. Ruland
expressor software corporation

February 2008





As business requirements continue to accelerate, the ability of existing data integration technologies to solve those requirements lags appreciably behind. In an effort to address this issue, vendors have been frantically retooling their platforms and/or buying “complementary technologies” to help them stay afloat. Unfortunately, these incremental improvements and bolt-on solutions have not been successful in resolving the core problem. What is needed is a fundamental shift in the way that data integration projects are approached.

Broken is the ability of the current crop of data integration platforms to capture and adapt to the nuances of our environments. These platforms were revolutionary when they were released, from pioneering the data mart architecture to the introduction of true parallel data movement. But the foundations of these data integration platforms are long of tooth now and struggling to keep up with the accelerating pace of today’s business.

We are in need of a new revolution that provides a way to isolate the business rules (the foundation of our applications) from the constant churn of new sources, new delivery mechanisms and new requirements. We are nearing a point where low latency will not only describe the data we are processing, but also our ability to rapidly respond to changing business initiatives. At the same time we need to prevent “baking” these rules into the application in a way that makes them brittle and stifles reuse. No data integration platform to date has even come close to doing this successfully.

semantic discord

Abbott and Costello’s signature routine – “Who’s on First?” – demonstrated not only the humor but the potential profitability to be found in carefully crafted, semantic word play. Anheuser-Busch recently illustrated that a single word – “dude” – can have a myriad of meanings depending entirely on context, tone or inflection.

Unfortunately, in the business world misconstrued meaning is usually unintentional and rarely a laughing matter. Like the childhood game of “telephone,” where simple sentences are whispered from one participant to the next, companies may not identify their mistakes until the game is over.

Multilingual confusion can derail the best-laid plans of multinational corporations, where simply translating a message from one language to another can leave ruined reputations and profit margins in its wake.¹ But frequently, even miscues with everyday business terminology can undermine successful outcomes. Say, for example, the boss states: “I want a good ROI.” What kind of return on investment does she seek? One calculated on the NPV or the IRR? Or is it a much more basic payback-period formula that she wants applied?² Maybe she is not really talking about a financial payback formula at all, but instead she means, “provide currency-based investment payoffs,” “do a business case to evaluate the scenario,” or even “don’t waste resources on performing a business case.”



Given the difficulties inherent in communication between humans, it comes as no surprise that the problem is compounded when extended to the IT realm. It is one of the major contributing factors underlying the difficulty in delivering successful data integration solutions in a timely fashion. It also stymies our ability to keep pace with the continual application modifications required in today's dynamic business environment.

For technologists, as well as linguists, the phenomenon is known as “*semantic discord*”; it refers to the way that the same data is called different names by those close to the business and those close to the data (i.e., the IT department).³

Semantic “disputes” of this sort are said to be “*semantically loaded*”,⁴ and many data integration efforts could be said to be semantically overloaded.

historical roots

There are many technical reasons responsible for this situation. Historically, the physical name associated with a data item had limitations as to how long it could be: databases were limited to 18-character names and COBOL programs were limited to 30-character names.

Esoteric engineering was required by the IT department to encode considerable information into this limited space. For example, a coded value for the application associated with the data might be included along with abbreviations for the contents – occasionally resulting in absurd, obscure names. (How apparent are the references embedded in names like OD-BO-TRAN or DSH-DOSYANUM-R?) These cryptic names present hurdles for implementation teams new to any environment: learning the language of the physical data is different in almost every situation.

To make matters worse, there are frequently multiple sources for the same data item, each with a *different* encoded name – another manifestation of semantic discord, where different terms refer to the same item. Furthermore, these different source names may be stored in a dissimilar physical format.

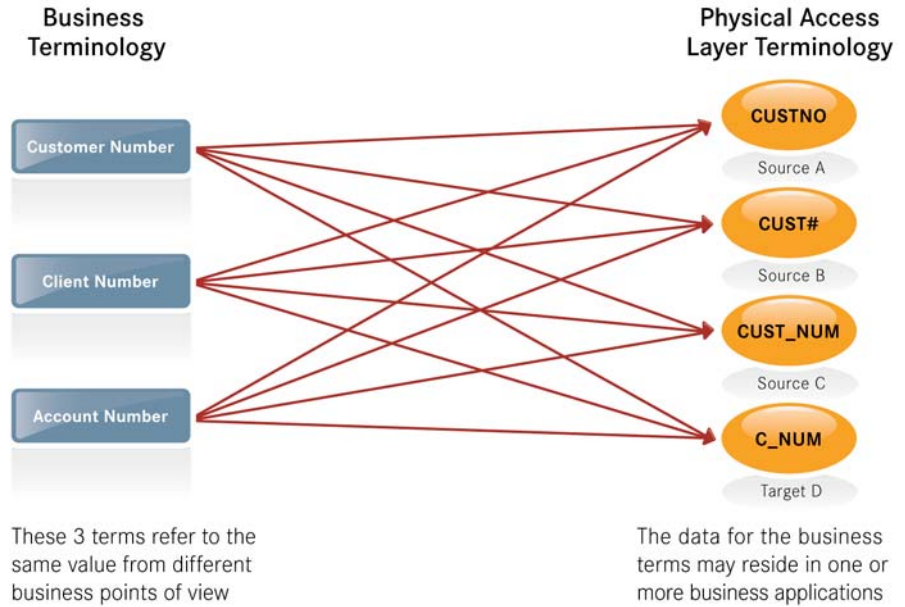
Consider the infamous case where a critical component in the space program was incorrectly translated from English units to metric units, causing the Mars Climate Orbiter (MCO) mission to fail and prompting the restructure of the NASA project management process.⁵

So IT often faces multiple, encoded data-naming conventions, with which even the implementation teams can struggle. This is one of the all-too-familiar challenges associated with maintenance and reuse. Even if current technology supported restating all the metadata in the millions of COBOL copybooks and legacy databases, the conversion task would be massive and extremely risky given the coordination and accommodation of inter-relationships it would require. Clearly, the solution lies in the adoption of a different approach.

Combine the vagaries of differing semantic interpretations with cryptic legacy naming conventions, and the situation becomes fundamentally unmanageable and untenable. This ever-widening gap between the multiple business meanings and the multiple technical meanings and definitions can grow at a cancerous rate, crippling opportunities for sound fiscal growth.

Figure 1 illustrates the complexities that can surface when even three names referring to the same business items (Customer Number, Client Number and Account Number) are encoded differently in four different physical structures. Imagine the complexities that surface with even hundreds of business terms and dozens of sources and targets. How many business terms, sources and targets does your business deal with?

figure 1: the concept of semantic discord: as terms are added, complexity compounds



the “6X spend”

A costly consequence of the semantic load shouldered by data integration initiatives is the “6X Spend” phenomenon.⁶ Plainly stated, for every dollar spent on a software/hardware solution, businesses end up forking over another \$6 to implement it. The reasons for this inflationary spending range from training to managing and marshalling data. Often companies end up with dozens, if not hundreds, of spreadsheet-toting analysts reconciling data and business rules caught in the semantic discord chasm. In fact, this proliferation of data and technologies has given rise to the discipline of Master Data Management.

Given the pricey nature of data integration solutions, an initial investment in the technology might carry a price tag in the neighborhood of \$1 million; given the discord surrounding the semantics of business users and data curators, 6X can jump up to 10X; and finally, given the propensity for failure of those efforts,⁷ the bill may escalate even higher before anything actually delivers business value.

You can almost hear Bud Abbott screaming “Duuuude!!!!!!” instead of “Looooouu!”

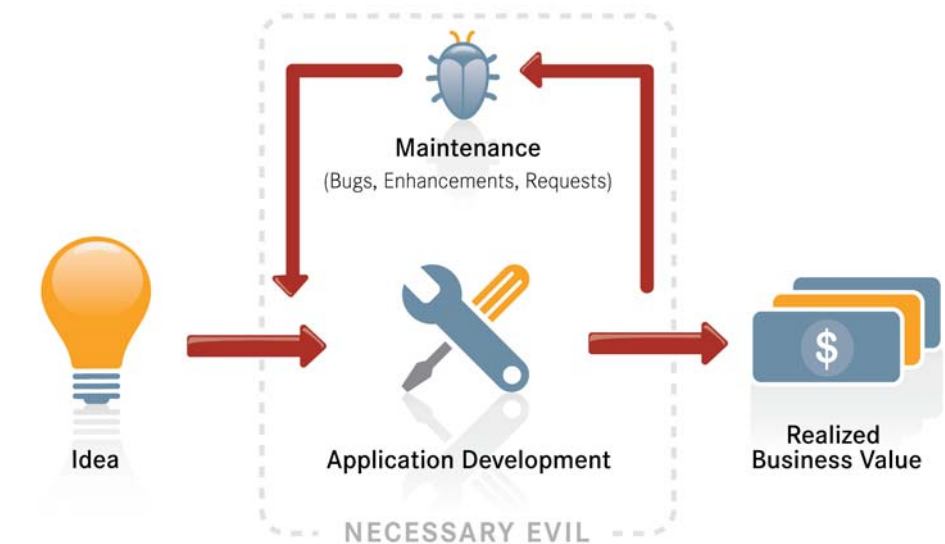
Above and beyond these factors, semantic discord introduces a major obstacle for companies who need to adhere to the plethora of compliancy requirements common to the business world. Unfortunately, these requirements are destined to increase in

the future so our costs will undoubtedly compound over time. This situation will be increasingly challenging for organizations as data sources, volumes, and applications grow.

semantic discord and the development cycle

Let's now look at the impact semantic discord has on the various stages in the lifecycle of an application. Our goal here is to survey the application development lifecycle at a high enough level that we can keep the big picture in view. **Figure 2** shows a simplistic view of the application life cycle from the point of view of the business. The objective of the business is to shorten the time from idea to realized value. From this view, it is obvious that we can address the goal of reducing the impedance between these two positions by improving our development speed and reducing our errors. This "revelation" should surprise no one. At issue is how to accomplish this lofty goal – after all, everyone seems to claim that they provide it.

figure 2: the business view of application development



how semantic discord impacts project startup

Dwight D. Eisenhower said, "Plans are nothing; planning is everything." Accurately determining the business requirements is unquestionably among the most important components to a successful project – and, of course, the solution must be delivered within a timeframe that is useful to the business. There are many different methodologies available to structure the organization of a project, from waterfall to agile methods. These methodologies have been developed over time to manage and expedite application development and delivery. The tasks and concepts covered in this portion of the discussion are not specific to any particular methodology, but are typical activities conducted in various degrees by all methodologies. Some of this activity may be pushed into the development process with the agile-based methodologies and resurface in concepts such as use cases, storybooks, and scenarios. Regardless of the chosen organizational structure, a considerable number of participants are involved and the requirements may pass through a number of different tools and translations, each hampered by semantic discord.

One of the first things to occur in the life of most applications is that someone has an epiphany about how technology could be applied to improve the business. Following this idea through the planning phase reveals that the original concept is interpreted multiple times by individuals in different roles and by encoding in different tools. Each of the activities shown in **figure 3** opens the door to semantic discord. Worse than that, many of these activities are begun using information created in a prior activity. For example, the physical database structure is typically built from a data model, which was built from a business model, etc. Does this begin to sound dangerously close to our game of “telephone?”

figure 3: planning a project

Activity	Role(s)	Example tool(s)
• Business case creation	• Business analyst • Architect	• Microsoft Word • Excel • PowerPoint
• Business/data model creation	• Data analyst • Modeler	• CA ERwin • Rational Data Architect • Sybase PowerDesigner
• Project plan creation	• Project manager • Architect	• Microsoft Project • Excel
• Data source identification	• Data analyst • SME	• Data profiling tool • SQL
• Physical target creation	• DBA	• SQL GUI tool • SQL
• Mapping specification creation	• Business analyst • Architect • SME	• Microsoft Excel

One of the activities, creation of the data model, contains a clever attempt to address the semantic discord problem by introducing the concept of a logical and a physical data model. The business terms (logical model) are mapped to the physical names and data structures (physical model) stipulated in the external storage media to provide a semantic bridge between the two disciplines. Unfortunately, in the development phase of the application the implementation team still needs to use all the diverse physical names, reintroducing semantic discord into the process. As we will see in the development section of the discussion, this reintroduction wipes out much of the benefit of the logical/physical model for the development process.

Another activity that deserves special attention is the creation of the mapping specification, which defines the business rules to the implementation team. These rules are originally captured in word-processing documents by a business analyst and generally are not directly applicable to creating the application. The normal “solution” is for an

architect, business analyst or subject matter expert (SME) to create a mapping document, typically in a spreadsheet. The vocabulary in the mapping document is generally a “technical” pseudo language. There is no widespread standard for this pseudo language, so it commonly ends up looking slightly different depending on the specification’s author. Another curious thing often happens during creation of the specification mapping: Although focused on building a target structure to resolve a business need, we create a source-to-target mapping rather than a target-from-source mapping. This might seem like a subtle difference, but it can make specification of the business rules very confusing and significantly increase the opportunity for mistakes. Many experienced organizations realize this and build all their mapping specifications from the target viewpoint.

Obviously, there are a considerable number of participants, tools and moving parts involved in developing an application even before the implementation team begins to code anything. After considering the effort involved, newcomers to the development of a large-scale application are often overwhelmed and have been heard to say, “No wonder these projects fail so often.” Modern methodologies attempt to reduce some of this pre-development work and move it closer to the development effort. This introduces feedback mechanisms, such as frequent review by the business users, to try to help manage the infusion of semantic discord.

how semantic discord impacts project development

With all the complexities we’ve discussed related to project initiation, the actual development of the application may seem a little anticlimactic. The first struggle that the implementation team faces (especially in remote development situations) is to decipher the specifications assembled in the mapping document (i.e., the spreadsheet). As we mentioned, the pseudo code used in the specification is typically not formalized and may be different from other specifications due to the nuances introduced by the document’s creator. Decoding these rules becomes more challenging in correlation to the complexity of the mapping rule. Quite often errors are introduced because of misunderstandings in these specifications.

Another shortcoming of the specifications is that the rules are often written without a good understanding of the data involved in the transformation, a situation that can occur even when data profiling is done. Failure to validate the rules against the data while constructing the mapping specification forces this testing to be done by the implementation team. However, the implementation team doesn’t necessarily know the business application rules; they are often left on their own to make decisions (consciously or not) about the implementation details. Making decisions at this juncture potentially skews the implementation rules from those documented in the specifications, confusing the semantic relationships even more.

One of IT’s goals is to maintain an accurate inventory of business rules and their relationships for future reuse. After all, reusing functionality is much more efficient than rebuilding it. Unfortunately, failure to reintroduce development changes back into the specifications may compromise our opportunities for reuse. This point is also important in order to avoid using an obsolete mapping specification and continually reinventing a rule correction, or reintroducing a previously corrected error into the next application. Recall that reducing errors was one of our objectives for addressing application development overhead.



The mapping specifications are typically written as source-to-target mappings and based on the physical names used in the source and target structures instead of business names. Because the physical names for the same business name may be different (depending on the source or target), we inhibit our ability to reuse rules or specifications associated with similar business elements – again, a manifestation of semantic discord.

how semantic discord impacts post-development activities

Because we have so many potentially different physical names for the same data element, it is exceedingly difficult to both find and reuse logic unless the physical names and data formats are identical in the sources and targets. This is one of the primary reasons

that reusability and maintainability promised by application development platforms seems out of reach. Semantic discord effectively obfuscates our inventory of business rules, often requiring us to rebuild a rule when it is applied to a discordant external name and hiding the fact that a similar rule may have been used elsewhere. Achieving semantic harmony within the development environment is necessary in order to resolve this effect.

A related problem we face is streamlining application maintenance with today's data integration technologies. Since our ability to reuse logic is impaired by the use of different physical names, we are not able to adequately find and eliminate redundancies or inconsistencies in the transformation logic within our applications. The hardcoded manner in which the transformation logic is encoded into the data integration applications inhibits our ability to easily customize the application or react quickly to business demands. This is the key reason many users are distrustful of the reusability and maintenance claims made by data integration software vendors.

summary

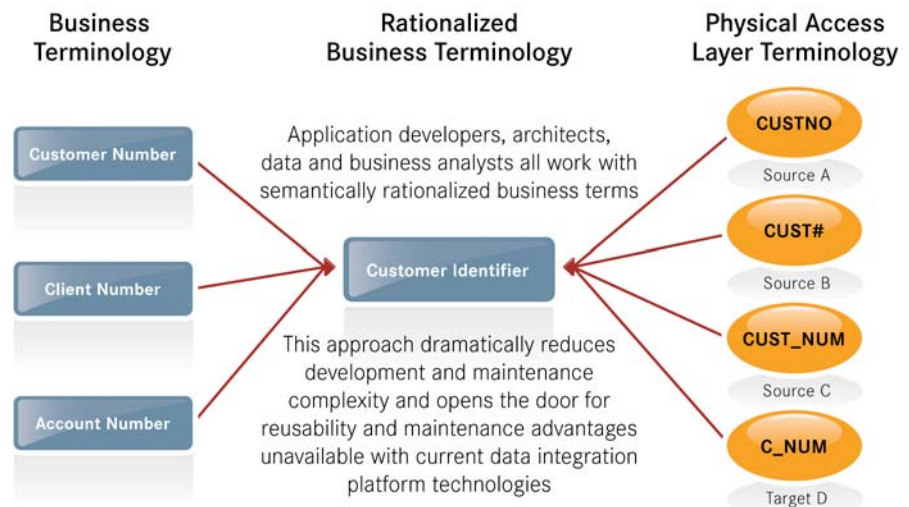
We have briefly explored the concept of semantic discord and seen how its effects ripple throughout the data integration application development process. Contrary to the playful way that Abbot, Costello and Budweiser have portrayed the humor in semantics, semantic discord is causing a very real and serious problem for the IT industry today. Solving the associated problems is not trivial, and many organizations are experienced combatants in the ongoing struggle. The typical vendor "solution" involves creation of a "simple" business glossary to which users and technicians refer for reconciliation of business and technical terms. Business glossaries are roughly similar in concept to the logical/physical data model solution discussed earlier. While this approach may help, it is still a manual process that only provides a tenuous bridge between the two reference domains, without really addressing the underlying problem. Business and technical communities need to use the same terms, not just be able to traverse between the vocabularies in a multilingual fashion. In other words, these glossaries are not pervasive into the development process; the implementation team continues to work with the cryptic names that likely change from one enterprise or application to the next. Other niche vendors are attempting to resolve the issue across diverse technologies such as COBOL, Java, and ETL tools; this is proving to be an extremely complex problem and the intricacies involved in implementing solutions have resulted in limited, costly, and often unwieldy solutions.

expressor software has changed the development paradigm so that the external names used in the physical storage media are rationalized once to well-defined, meaningful names that are then used by the business stewards and technical areas to refer to

common data elements. This concept is illustrated in **figure 4**. Since these rationalized names are common throughout the process, consistent semantics and reuse are automatic byproducts of the architecture. By coupling business rules with the rationalized names, and not “baking” them into the application, the platform also supports building applications that are far less brittle.

Companies are losing money, out of compliance, and running out of time. CEOs and CFOs are losing their jobs because their business insight depends on brittle IT solutions that seem to take forever to implement. Dramatic changes are needed on the approach to data integration. The confusion and paralyzing effects of semantic discord must go. And business needs must be supported now, not later.

figure 4: breaking out of the semantic discord dilemma



footnotes

- ¹ for an anonymous compendium of translation mishaps, see: http://baetzler.de/humor/ads_gone_wrong.html
- ² net present value (NPV) is a standard method for the financial appraisal of long-term projects. It measures the excess or shortfall of cash flows, in present value (PV) terms, once financing charges are met. The internal rate of return (IRR) is a capital budgeting metric used by firms to decide whether they should make investments. Payback period refers to the period of time required for the return on an investment to “repay” the sum of the original investment.
- ³ conversely, differing data may share the same name...more on that momentarily.
- ⁴ http://en.wikipedia.org/wiki/Semantic_discord
- ⁵ <http://mars.jpl.nasa.gov/msp98/news/mco991110.html>
- ⁶ an Ovum report introduced this concept
- ⁷ *A 50% Data Warehouse Failure Rate is Nothing New*, <http://blogs.ittoolbox.com/eai/rationality/archives/a-50-data-warehouse-failure-rate-is-nothing-new-4669>



Michael Ruland has more than 30 years of IT experience, including more than a dozen focused on constructing and maintaining Data Warehouse and Decision Support Structures. He has worked as a global technical architect for IBM, in addition to application programming, systems programming, and various sales and engineering positions with Amdahl, Informix, Prism Solutions, Ardent and Ascential.

expressor software corporation provides a next-generation data integration platform built to address the most difficult problems facing organizations involved in data integration today.

For further information please contact us at:



1 new england executive park
burlington, ma 01803
USA
+1 781.505.4190
+1 781.505.4191 fax

www.expressor-software.com

Copyright © 2008 expressor software corporation. expressor, expressor semantic data integration system, smart semantics, and redefining data integration are trademarks of expressor software corporation. All other trademarks or trade names are properties of their respective owners. All rights reserved.

PRIVATE AND CONFIDENTIAL