# An Ontology-Based Software Agent System Case Study[1]

Frederick T. Sheldon[2] Mark T. Elmore and Thomas E. Potok
*Applied Software Engineering Research Group*
Computational Sciences and Engineering, Oak Ridge National Lab[3]

## Abstract

This case study demonstrates a flexible and dynamic approach for fusing data across combinations of participating heterogeneous sources to maximize knowledge sharing. Software agents are used to generate the largest intersection of shared data across any selected data source subset. This ontology-based agent approach maximizes knowledge sharing by dynamically generating common ontologies for the data sources of interest.

We validated our approach using (disparate) data sets provided by five national laboratories. A local ontology was defined for each laboratory data source. The ontologies specify how to format the data using XML to make it suitable for query. Consequently, software agents are empowered to provide the ability to dynamically form local ontologies from the data sources. In this way, the cost of developing these ontologies is reduced while providing the broadest possible access to available data sources.

## 1. Introduction

Developing a knowledge-sharing capability across different distributed data sources continues to be a considerable challenge. Ontology-based approaches show promise by resolving heterogeneity, if the participating data owners agree to use a common ontology (i.e., mapping among a set of common or similar qualities). Such common ontologies enable distributed data to be virtually located in a central repository. This knowledge sharing may be achieved by determining the *intersection* of similar concepts (i.e., qualities) from across various heterogeneous systems. However, if information is sought from a subset of the participating data sources, there may be concepts common to the subsets that are not included in the full common ontology, and therefore cannot be shared. To overcome this dilemma, we construct a series of ontologies, one for each possible combination of data sources, which map the different qualities that agreeable correspond. No concepts are lost, but the number of possible subsets is prohibitively large.

Each Department of Energy (DOE) national laboratory has evolved a unique business model for managing research proposals over the past sixty years. Given the historical nature of these evolutions, both the business models, and their associated (heterogeneous) data collections, are deeply rooted. A way was needed to merge data from the various systems as if the data was gathered and stored in a centralized repository.

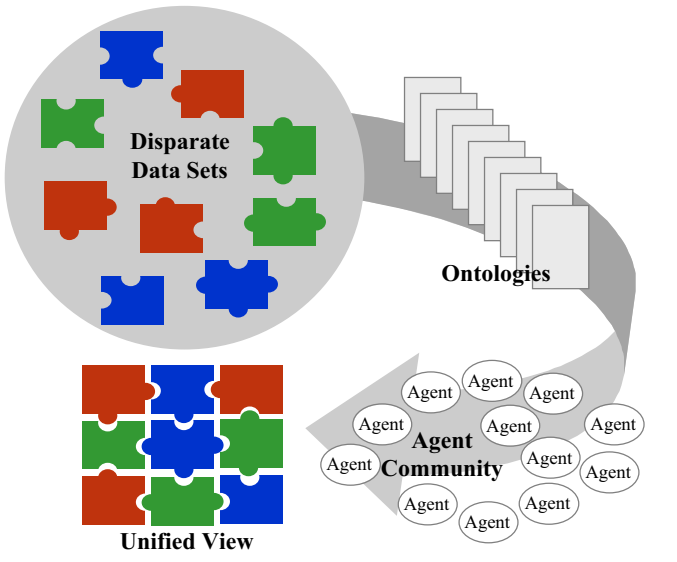


Figure 1. Ontologies and agents organize heterogeneous data.

Figure 1 (upper left hand corner) illustrates how the different labs within DOE have a somewhat different collection of research proposal data. Each collection and its corresponding ontology are based on the unique evolution of how each separate lab used the data within their respective business models. Moving to a single common ontology to resolve data heterogeneity across all laboratories will result in an intersection of all data concepts. Unfortunately, only common concepts are retained. Moreover, if two or more labs have a

[1] A different version of this paper, which discusses the issue of labeling entitled *Dynamic Data Fusion Using An Ontology-Based Software Agent System*, will appear in a special session of Agent-Based Computing 2003 (ABC'03) in conjunction with The 7th World Multiconference on Systemics, Cybernetics and Informatics.

[2] Contact: Post Office Box 2008, MS 6363, Oak Ridge, TN 37831-6363, Phone: 865-576-1339, Fax: 865-574-6275, SheldonFT@ornl.gov.

[3] This manuscript has been authored by a contractor of the U.S. Government under contract No. DE-AC05-96OR22464. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government Purposes.

concept that is not common, then (potentially) valuable information is not being represented. In this case, a common ontology that results in loss of such data has an unacceptable impact on the associated business models.

Figure 2 shows how concepts are lost in a common ontology. Consider three data sets labeled *A*, *B*, and *C* each of which contain three concepts, *Lab Name*, *PI Name*, and one other that is not common. Lets examine the merged data from data sets A and B. The first two concepts match across all three but the third concept is lost. This is unfortunate because we are concerned only about data from sets A and B (see Figure 2 right lower corner). A better method is needed that would alleviate this unfortunate problem.

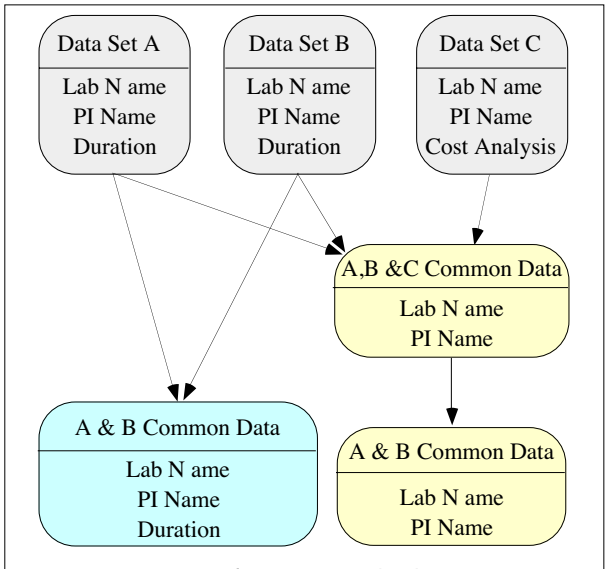

Figure 2 Example of common and subset ontologies.

In this example, we need to add *Duration* to the ontology when only data sets A and B are considered. If possible, then full visibility of all data across all combinations of participating data sets could be accomplished with a *series* of merged ontologies. The ontologies are merged in a way that provides for each possible combination of source data concepts, as compared to a single common ontology. Alternatively, there are a large number of possible combinations a user may choose potentially generating a huge number of ontology combinations. Current ontological approaches for merging heterogeneous data sets have been successful, but require *all* data owners to participate in building a single common view. For this reason (see Section 10, Appendix A), the large number ontologies for every possibility cannot be "pre-built." Instead, we propose to use software agents to build the desired ontology combination on-the-fly for each user-generated merge and query/search operation (as shown in Figure 1 bottom half). In this way only relevant merges are implemented, avoiding the need to generate all possible combinations while satisfying the possibility for merges that consider all possible combinations (including the intersection of every data set).

## 2. Background

An early use of the term "ontology" was in philosophy (Wolffe, C., Philosophia Prima Sive Ontologia, 1729) where its use means "a branch of metaphysics relating to the nature and relations of being." Computational scientist's adopted the general meaning of the term to be the "relations of being" for describing how data is related. Ontologies have proven to be a useful tool for data integration across heterogeneous data sets.

In effect, ontology specifies a vocabulary that is used to discuss a problem domain. For example, an ontology for baseball would include terms such as ball, bat, glove, strike, foul, etc. But it is not so much what terms are *used*, but what those terms *mean* that reaches to the core of how ontologies are used. For example, changing the language of the ontology from English to French changes the terms used, but does not change the concepts specified by the terms [1]. Gruber sums this up well when he describes ontology as an "explicit specification of a conceptualization" [2].

Cross-platform exchange of information increases the demand for a uniform view of related sources. Most approaches employ an agent to capture data semantics (the information content of each source [3]) using descriptive, domain independent, and semantic metadata. Metadata descriptions are used to resolve schematic and structural differences [4] and are constructed using terms, which stem from a given vocabulary, domain, or common sense ontology, like Cyc or WordNet [5]. This type of conceptualization of real-world notions reusable across shared domains is the key to reconciling semantic differences. Tools for building such ontologies include standard modeling and markup languages such as UML [6], XML [7], RDF [8], ontology/conceptual knowledge markup language (OML/CKML), SHOE (Simple HTML extension for Web page annotation), XML-based ontology exchange languages OIL and XOL, and description logics [9, 10].

In several revisions from 1993 to 1995, Gruber [2] represents a foundational set of design criteria to guide the development of ontologies in support of knowledge sharing activities. Gruber applies formal engineering discipline to ontology design using a core set of five design criteria as follows. The ontology should provide *clarity* in defining terms and *coherence* by being logically consistent. It should provide *extendibility* to allow expansion without affecting existing definitions, and should have *minimal encoding bias* so that the notation used to describe a concept does not restrict alternative ways to understand the concept. Finally, an ontology should have *Minimal ontological commitment*, meaning that the description of concepts should be as loose as possible to permit flexible use of the described concepts. These seminal design criteria form the basis for a series of ontological studies.

Recent in this series, Holsapple and Joshi [11] adopt Gruber's view and give a taxonomy of five approaches to Gruber's ontology design criteria. First lets consider the *inspirational approach*. In this case, an individual uses their own insights and viewpoint to develop an ontology of the domain that is then (hopefully) adopted by other users. The *inductive approach* builds an ontology based on a specific (set of) case(s) within the domain and is refined by evolving toward a more generalized ontology. The *seductive approach* moves in the opposite direction, beginning with an ontology built upon general principles of the domain and evolving toward fulfilling specific cases. The *synthetic approach* is a combination of existing ontologies into a single all-encompassing ontology that describes the combined domain. Finally, in the *collaborative approach*, a team of individuals incorporates aspects of the other approaches to build an ontology using the combined viewpoints and (possibly) using existing ontologies as an anchor. Holsapple and Joshi find the collaborative approach the most useful for their problem, and present a case study.

These various ontology based approaches, from Gruber to Holsapple and Joshi, provide methods to discover the homogeneity that may be found among heterogeneous data sets, and from that, build a common ontology. However, these approaches assume that participants are capable of migrating their data to a new ontology; this was not the case in our problem. The varied data collections within DOE are tightly coupled to their associated business processes. So much so, that it became unclear whether the business model drives the data or the data drives the business model. Migrating the data to a common ontology would necessitate a prohibitively expensive change to long and well-established business models. Thus, these powerful approaches are not suited to our problem without a variation from previously seen ontology-based data fusion.

## 3. Our approach

Given the aforementioned ontological approaches to data merging we decided that the power of these tools showed great promise. Current ontological approaches to the merging of heterogeneous data have been successful, but require the owners of the data to participate in the adoption of a single common ontology. In our case, we cannot go to a single, all encompassing, common ontology because of the importance of concepts occurring in most, but not all data sources. The solution to this would be to build a series of ontologies for all possible combinations of the underlying data sets. However, in our case this is an intractable problem and we decided to explore the possibility of using software agents to perform the ontology building tasks automatically.

Foundational to our approach is the use of the Extensible Markup Language (XML) as the mechanism for capturing data [7]. Each laboratory has a different mechanism for capturing data, from databases, to spreadsheets, to ad hoc text files and various combinations. Using a series of specialized software tools, the various data formats are converted into XML. XML provided an efficient mechanism to bring the distributed and heterogeneous data formats into a powerful, flexible, and common format thereby providing a standard ontology-to-software agent interface [12]. Consequently abstracting unnecessary details from the underlying datasets proves to be a sufficiently rich environment for the software agents to perform the merging of ontologies.

Next, we designed the coordinator for the system, a mechanism that would allow software agents to understand the various data sets and enable the agents to merge ontologies. While formal ontologies have a great many strengths, one potent drawback is the learning curve associated with using them. In our problem, the data owners who understood the data did not have the time for such a learning curve. To address this conflict, we devised a simple system that specifies data concepts by defining the equivalence relations between a data owner's local data concepts and other participating data concepts. Our technique works in the same way that the concept for "table" can be specified by showing equivalence to an already understood concept of the Spanish word "mesa," and/or the French word "tableau," and/or the Italian word "tavolo," Although intentionally informal in its execution, the specification of the data concepts by their relationships meets Gruber's definition of an ontology.

For example, as shown in Figure 3, the first data set, *Data Owner A* provides a simple list of data elements. In a parallel column the list of elements is repeated. The first column is labeled *Local Data Concept List*; the second column is labeled *Master Data Concept List*. Together they represent a simple ontology, i.e., a specification of the data concepts represented by a mapping from the *Local Data Concept List* to the *Master Data Concept List*. Next we incorporate data from *Data Owner B* where data owners A and B know each other's data. This is the case across most of the DOE system; most data owners know their data as well as its relationship to a small number of the other data systems. This knowledge context provided the basis for constructing composite

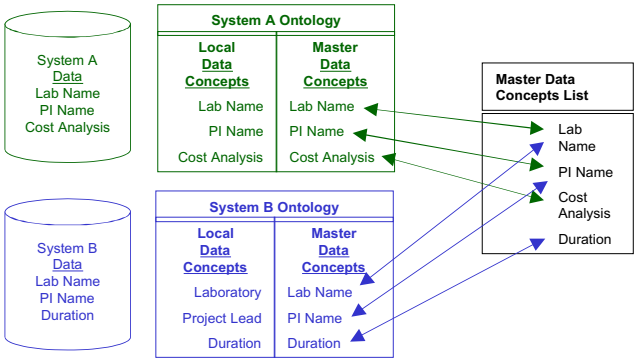

Figure 3 Software agents used to build ontologies.

ontologies (i.e., master data concept).

For the second data set, *Data Owner B* looked at *Data Owner A's* ontology mapping. *Data Owner B* could then provide us the knowledge needed to build a new ontology mapping, mapping B's data to the *Master Data Concept List*. Each data concept in a local data list was mapped to the same concept in the *Master Data Concept List*. If *Data Owner B* had data that was not in the *Master Data Concept List*, a new entry was added to the *Master Data Concept List*. Conversely, if a data concept in the *Master Data Concept List* was not present in the local ontology then there was no mapping established from his *Local Data Concept List* to the *Master Data Concept List*. In the end, the system consists of three elements: (1) *Master Data Concept List*, (2) *Local Data Concept List to Master Data Concept List Ontology* for Data Set A, and (3) *Local Data Concept List to Master Data Concept List Ontology* for Data Set B.

This process continues for each new data set. Each new data owner uses the previous work to help determine their ontology as a specification of the mapping from their *Local Data Concepts List* to the *Master Data Concepts List*. Questions pertaining to proper mappings are resolved (through discussion) between the new data owner and the data owner that previously added the data concept to the *Master Data Concepts List.*

Thus, in this process, the *Master Data Concepts List* is a union of the data concepts across all participating data sets, and a given data set's ontology is a mapping specifying the relationships between the intersection of that data set's local data concepts and the master data concepts. Relationships among a selection of the local data sets' ontologies can be determined using the *Master Data Concepts List* as a point of common reference. It is interesting to note that there is no centralized ontology for the entire system. Instead, it is distributed across the ontology mappings of the individual data sets and the *Master Data Concepts List*. For example, *Data Set A's* "PI Name" specifies the same concept as *Data Set B's* "Project Lead" but this cannot be directly determined at one centralized point; rather it is determined via the data set ontologies and the *Master Data Concepts List*. Software agents use this distributed ontology to provide the functionality of a centralized ontology along with the ability to be flexible in meeting the varied needs of the users.

As described above, we applied this approach on data from five national laboratories. These laboratories are very large and present massive data sets across a diverse set of repositories (e.g., databases, spreadsheets, and simple ASSCI files, etc.). We manually created the local ontologies, and used these ontologies to create XML representations of the data at the laboratories. From this base, we then applied our approach to building dynamic on-the-fly ontologies using agents.

## 4. Implementation Using Software Agents

For each data set, a *Data Agent* is assigned the task of retrieving data from the underlying data repository based on the defined ontology. The agent must present the data to the overall system for merging using the ontology defined for that particular *local* data set. The mapping provided by the particular ontology allows the data agent to understand both the local and master concepts. Based on the master data concepts side, the agent understands the language of the agent community; based on the local side, the agent understands how to retrieve local data. Ultimately, the agent uses this mapping to translate between the local and master. On the fly, a *Data Integration (DI) Agent* assembles merged ontologies, coordinating the activities of all the data agents in the community according to the user's requirements. This agent therefore accepts requests from a *Graphical User Interface (GUI) Agent*, another important type of bilingual agent, which translates user requests into the language used by the agent community. The results of the merging are subsequently derived into visualization for the user. The DI Agent distributes the GUI Agent's requests to the appropriate data agents, merges the results from the data agents, and passes the merged data back to the GUI Agent for visualization as shown in Figure 4. Missing agents cannot contribute data to a merging but do not prevent a merging from occurring. The agents and ontology approach provides an eloquent efficient and extensible solution to avoiding the combinatorial number of possible combinations.

## 5. Software agents build a common ontology

When a user of the system first brings up the GUI, the GUI Agent asks the Data Integration Agent for a list of available data sets, that is, a list of available data agents. The Data Integration Agent then checks the list of data agents that are registered, and verifies the availability of each. This agent then reports the availability to the GUI Agent, who displays the available data sources to the user. The user then selects the desired data sources and the software agents dynamically create a merged ontology for the selected data sources.

To create this merged ontology, the Data Integration Agent sequentially distributes the Master Data Concepts List to the data agents chosen by the user. The first data agent compares this concept list to his local ontology, and deletes from this list the data concepts that are not found (i.e., the data concepts that are not in the local ontology). The data agents then hands the reduced data concept list back to the Data Integration Agent who then passes the reduced list to the next data agent selected by the user. This process continues for each of the user-selected agents until all have seen the list.

The trimmed data concepts list resulting from this process is the intersection of the data concepts captured within the participating systems. Note how this process is the reverse of the way the Master Data Concepts List was

originally generated. In building the Master Data Concepts List, the data owners each *added* concepts from their local data sets that are new concepts to the Master Data Concepts List. Here, the data agents *remove* data concepts that are not part of their local ontology from a copy of the Master Data Concepts List

The final reduced data concepts list, in conjunction with each participating data agent's ontology, constitutes a shared ontology across the participating data sets. This ontology is dynamically generated based on a request from a user, and is evaluated against the latest information from each local data source. Participating agents can each understand and provide information about all the data concepts that are shared across the participating systems, which significantly increase the capability of current ontologies. Moreover, the full system addresses Gruber's five ontology design criteria. Using relationships to specify concepts provides *Clarity* and *Coherence*. *Extendibility* has been shown in the process by adding new data sets. And the succinct specification renders *Minimal Encoding Bias* and *Minimal Ontological Commitment*. This process could also be thought of as the software agent version of Holsapple and Joshi's *collaborative approach* to ontology design, producing results in much the same way a group of collaborating humans would have done, but significantly faster, and with far greater accuracy.

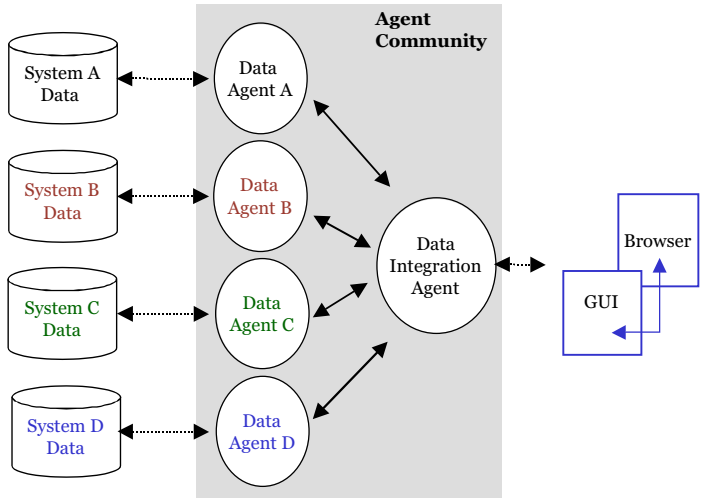

Figure 4. Agents are used to integrate various ontologies

## 6. Querying over merged data

The GUI offers the human user an interface to specify a query over distributed data as if it were collected in one location under a single schema. Our version of the GUI was designed to provide the user with rich query capabilities to permit users to specify down to the data fields of interest (as in an SQL *select*) and, at the same time, permitting constraints on data field values (as in an SQL *where*).

Queries are passed to the Data Integration Agent who partitions the query out to the selected set of data agents. Agents are then responsible to fulfill the query over their local data set. The underlying XML data set may be stored in any manner usable by the data agent. Each data agent uses the local ontology mapping to convert the query request from the Data Integration Agent into a format usable against the respective data set. The answer to the query is then translated back into the format used by the software agent community and returned to the Data Integration Agent. For example, from Figure 3 a general search for a PI Name of "smith" would be translated to a search over a Project Lead with the name "smith" within System B. Both the query and the response are XML format. The XML format enables the Data Integration Agent to assemble the individual responses from the data agents. The data integration agent then passes the assembled query response back to the GUI agent for presentation to the user.

## 7. Results

After completing the system design, five data owners were contacted and asked for their assistance by participating in a prototype demonstration. This selection was of sufficient size to test our approach; five data sets producing ($2^5$-1) or 31 possible combinations of merged data sets over which a user might wish to query (see Figure 5). This number is too large for the data owners to build all 31 possible ontologies; yet large enough to test the strength of our software agent enabled ontology-building approach.

The data owners were enthusiastic about the potential solution yet skeptical about using software agents as described above. In fact, all prior efforts with the same goal were unsuccessful. These data owners are extremely busy; they could only give small amounts of time to help validate our approach. Indeed, the approach proceeded better than expected. It took more time to explain the approach to the data owners than to actually incorporate their data. Some data took longer than others to incorporate (i.e., define in terms of a compliant ontology) because varied native formats, but none of the data took more than a couple of days.

The system performed without a glitch. Acquisition times for query results were negligible, with network latency being the bottleneck. Delays were similar to downloading a web page of typical complexity, well within most typical users' tolerances for delay. The sponsor deemed the prototype a success. Due to the success of the prototype, the DOE decided to implement the system across all of its installations, estimating a cost savings of $39 million per year. Consequently, the DOE has now begun the process of building a production system based on our prototype.

## 8. Conclusions and future work

Ontology based merging of data has proven to be a viable technology in a great many instances, but not all problem domains succumb to these techniques. We have described a problem domain where data owners have data that they wish to share, but they cannot move to a single common ontology because of the potential loss of information (incoherency). One inadequate approach uses a series of ontologies for all possible (i.e., brut-force) combinations of data, but is prohibitively expensive.

Using our approach we have demonstrated the use of software agents to dynamically create merged ontologies, which significantly reduce the cost of developing brut-force ontologies, while providing the broadest access to distributed information. These ontologies meet the requirements stated by Gruber and others producing a shared ontology across the participating data sets. In this approach, ontologies are dynamically generated based on a user request that is evaluated against the latest ontological information from each local data source. Participating agents can each understand and provide information about all the data concepts that are shared across the participating systems, which significantly increases the capability of current (i.e., latest agent derived) ontologies. The implementation of this approach produced significant financial benefit, and will soon see broad deployment.

One idea that kept occurring as each new data set was added was the automation of ontology production. The idea would be that software would make an initial draft of a new data set's ontology. The time required to generate a final draft of the ontology would then be reduced, requiring only a final editing and/or verification by the data owner. We envision building ontologies by using clustering techniques; new data concepts would naturally cluster close to matching data concepts that have already been incorporated.

### How Many Possible Combinations?

When we wish to determine the number of combinations of k objects chosen from n objects, where order does not matter, we say n choose k, or C(n,k):

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

To determine all possible combinations, we desire the summation of C(n,k) for all values of k from 1 to n:

$$\textit{Possible Combinations} = \sum_{k=1}^{n} \frac{n!}{k!(n-k)!}$$

For n=520 :

$$\textit{Possible Combinations} = \sum_{k=1}^{520} \frac{520!}{k!(520-k)!}$$

**Pascal's Triangle**

| | | Total Combinations Per Row |
|---|---|---|
| Row 0 | 1 | 1 ($2^0$) |
| Row 1 | 1 1 | 2 ($2^1$) |
| Row 2 | 1 2 1 | 4 ($2^2$) |
| Row 3 | 1 3 3 1 | 8 ($2^3$) |
| Row 4 | 1 4 6 4 1 | 16 ($2^4$) |
| Row 5 | 1 5 10 10 5 1 | 32 ($2^5$) |
| Row 6 | 1 6 15 20 15 6 1 | 64 ($2^6$) |
| Row 7 | 1 7 21 35 35 21 7 1 | 128 ($2^7$) |
| Row 8 | 1 8 28 56 70 56 28 8 1 | 256 ($2^8$) |
| Row 9 | 1 9 36 84 126 126 84 36 9 1 | 512 ($2^9$) |
| Row 10 | 1 10 45 120 210 252 210 120 45 10 1 | 1024 ($2^{10}$) |

A simple way to determine all possible combinations for large numbers is to use Pascal's Triangle. The 520th row of Pascal's Triangle shows all possible combinations, from k=0 to k=520. The zeroth entry will show the number of possibilities for k=0, the first entry will show the number of possibilities for k=1, and so on.

As seen in the beginning sequence of Pascal's Triangle, above, the total number of possible combinations for any non-negative integer n is $2^n$. But since we are not interested in 520 things taken 0 at a time (520th row, 0th entry = 1) we subtract 1 from $2^{520}$.

Therefore, the total number of possible combinations from 520 datasets = $2^{520}$-1, or just a "little" under $3.5 \times 10^{156}$.
And the total number of possible combinations from 20 datasets = $2^{20}$-1 = 1,048,575.

Figure 5. Pascal's Triangle illustrates the combinatorics of merging data sets.

Full visibility of all data across all combinations of the underlying data sets could be accomplished with a series of merged ontologies, one for each possible combination. However, there are a large number of possible combinations a user may select from the underlying data sets. With twenty-six DOE laboratories, and perhaps twenty different systems to be merged, there could be as many as 520 different data sets if each laboratory has a unique variation on every system. In reality, there is not uniqueness at every possibility. Still, 520 can serve as an upper bound and perhaps 20 can serve as a very optimistic lower bound. This means the number of possible combinations a user may wish to select would be between an upper bound around $3.5 \times 10^{156}$ and a lower bound of 1,048,575 – still a prohibitively large number of possible combinations.

## 9. References


1. Chandrasekaran, B., J.R. Josephson, and V.R. Benjamin, *What are Ontologies and Why Do We Need Them?* Intelligent Systems, 1999. **14**(1): p. 20-26.
2. Gruber, T., *A Translation Approach to Portable Ontology Specifications.* Knowledge Acquisition, 1993. **5**(2): p. 199-220.
3. Klusch, M., *Information Agent Technology for the Internet: A Survey.* Journal on Data and Knowledge Engineering, 2001. **36**(3).
4. Kashyap, V. and A. Sheth, eds. *Semantic heterogeneity in global information systems: the role of metadata context and ontology*. Cooperative INformation Systems: Trends and Directions, ed. M.P. Papazoglou and G. Schlageter. 1998, Academic Press, New York.
5. WordNet, *A lexical database for English*. 2001, http://www.cogsci.princeton.edu/~wn/.
6. Fowler, M., *UML Distilled: Applying the Standard Object Modeling Language*. 1997: Addison-Wesley, Reading, MA.
7. W3C-Recommendation, *Extensible Markup Language (XML) 1.0 1996-2003 (c)*. 1998, World Wide Web Consortium http://www.w3.org/XML.
8. W3C, *RDF Vocabulary Description Language 1.0: RDF Schema*. 2003 (c), World Wide Web Consortium http://www.w3.org/TR/rdf-schema/.
9. Buchheit, M., et al., *A refined architecture for terminological systems: terminology = schemas + views.* Artificial Intelligence Journal, 1998. **99**(2).
10. Donini, F.M., et al., *The complexity of concept languages.* Information and Computation, 1997. **134**(1).
11. Holsapple, C.W. and K.D. Joshi, *A Collaborative Approach to Ontology Design.* Communications of the ACM, 2002. **35**: p. 42-47.
12. Potok, T.E., M.T. Elmore, and N. Ivezic. *Collaborative Management Environment.* in *InForum*. 1999. Washington DC: Office of Scientific and Technical Information (OSTI) – www.osti.gov and Department of Energy (DOE) - www.energy.gov.