An Algebra for Ontology Composition

Gio Wiederhold

ARPA and Stanford University
July 1994

INTRODUCTION

To compose large scale software there has to be agreement about the terms, since our models depend on symbolic linkages among the components. In modestly-sized systems, we implicitely count on such an agreement. Within a specific domain terms are indeed likely to be consistemt, so that specifications can be developed from English (or other natural) language source documents. When combined with a coherent framework we have the underpinnings for a Domain-Specific-Software Architecture (DSSA). In this abstract we propose extending concepts used in object-based structural algebras and DSSA research to a knowledge-based algebra, suitable for composing larger systems that span multiple domains.

The principal operations in the algebras are simple and provide for selection from the objects in the source domain space and placing them into new domains that represent the information needed for the composed results.

1. BACKGROUND

Divide-and-conquer is an essential approach in science and technology, and in large software systems as well. Early manifestations of division of tasks in software were scientific subroutine libraries, since their development and evaluation required uncommon rigor. These libraries grew to encompass statistical procedures SAS [Rose:83], and specialized libraries serve diverse domains as planetary navigation NASA [Acton:93] and Graphical User Interfaces. Today commercial firms or funded service agencies provide most of such libraries.

An alternative approach is the development of packages, which are not intended to be integrated into larger suites. These were also popular in the statistical domain, as BMD [Dixon:69], but have been largely supplanted by composable routines, which allow the application of statistics to a variety of application domains.

In this abstract we consider the construction of software from autonomous modules, *megaprogramming* [Wiederhold:92]. We refer to the scope of autonomous modules as their domain, and to the terms used to describe items and their relationships in a domain as their domain `ontologies`. While the terms in these ontologies are often only manipulated in paper form or perhaps as IDEF [Loomis:87] documents, we believe that the contents of ontologies warrants formal manipulation if reliable systems are to be composed.

2. DOMAIN SPECIFIC KNOWLEDGE

Object technology has blosomed due to the incorporation of semantics within the units that the software and retrieval strategies deal with. The definitions that make retrieved objects coherent are particular to a specific application area and its domain. The focus of research in Domain-Specific-Software-Architecture (DSSA) has been the acquisition of knowledge in a specific *domain*. A working definition of a domain is an area of science or products where there is a common *ontology* Gruber:93].

Having a common ontology enables collaborators to work together with minimal risk of misunderstanding each other. When computer systems are used as the intermediaries in collaborative work, the need for a common ontology is even greater because many of the cues that exist in face-to-face interaction, the raised eyebrow, the wandering of attention, etc., cannot be perceived by one's partner.

The architectural aspect of a DSSA approach is that, once enough knowledge has been garnered about a specific domain, the object classes can be defined and placed in an operational relationship to each other [Haddock:94]. Within a domain, we assume consistency, namely, that the terms mean the same thing, i.e., refer to the identical object instances [Wiederhold:91], and have the same relationship.

3. DOMAIN DIFFERENCES

Having defined domains by their internal consistency, we must now consider the cases where such consistency does not hold. First of all, different domains will consider different objects. Different domains are likely to have different ontologies. These differences can be simply due to differences in naming and scope, both with respect to the names and semantics of meta-information about attributes that appear in the schema and the names and semantics that appear as values in the content of a database:

1 Naming attribute items differently. This is common, but also the easiest inconsistency to resolve. A example of this type occurs when employees are named in the `payroll` domain `EMP` and in the `personnel` domain `PEOPLE`. A simple table can be used to support the desired match and bring the information together.

2 Scope differences are much more insidious, and have to be determined by content analysis. The `personnel` domain may include assignees from other institutions, who are not listed in `payroll`. The `Payroll` may include support for student benefits for employee's children, but those children are, appropriatly, not listed in `personnel`. Resolution requires establishing, validating, and processing of rules. These rules can refer to variables in the domain that are not basic to the domain intersection.

3 Encoding differences of values are common as well. When numbers are used, a conversion can be established with a formula, say `meter = foot/0.305`. More complex are differences in dates and identifiers, say `ssn` with or without hyphens. Here rules have to be introduced as well, but when encodings are irregular, for instance `stock-codes`, tables have to be introduced. Tables dealing with instances of values occuring in databases require ongoing maintenance. We can hope that practical interoperation provides feedback which eventually will encourage coherence among domains.

4 Attribute scopes are often subjective. The term `hot` has a different meaning in the `weather` domain than the `truck-engine` or `truck-cargo` domains. If `hot weather` can effect the `truck-engine`, expert knowledge is needed to make the

linkage. Differences in scope lead to differences in referencing, which makes their resolution yet more critical. For example, both `patients` and `nurses` are subsets of `people`, but their roles in a hospital are quite distinct, so that it would be unwise to create generalized `people` objects and encapsulate all the differences internally.

No central organization can resolve all these differences, they require knowledge about the source domains and their intersection.

The differences enumerated above can make a once-and-for-all integration of distinct domain infeasible. A dynamic capability is essential if we wish to achieve associative access to multiple domains, since the transformations required to achieve optimization must maintain the correct semantics. For instance, the PENGUIN system constructs objects as needed out of relational databases, given a structural model of connecting references [Barsalou:91].

4. DOMAIN MERGING

There are several approaches to dealing with building composed ontologies from domains that have ontological differences:

1 Aggregate the terms from all relevant ontologies, give them to a committee, and ask them to prepare definitions that are acceptable to all. When the definitions are completely documented, release the document and expect that all participants will adjust their usage to conform to the definitions.

2 Assume that terms match, and when mismatches are discovered, make the terms distinct, typically by prefixing them with source or domain identifier. This is the approach used by UMLS [Humphreys:92]; all the sources are labeled to make such distinctions easy, and by CYC, where micro theories can encapsulate differences [Lenat:90]. Over time, the processes of sharing of information encouraged by the availability of the joint ontology will cause convergence of meanings, although coherence can never be assured.

3 Assume that terms never mean the same thing unless explicitly instructed. Such instructions, encoded as *matching rules*, form a knowledge-base to be managed by collaborators from two or more domains. No restrictions are imposed on the evolution of local terms within a domain. Terms that are covered by matching rules form a new, second layer abstract ontology. Higher abstract layers can be defined recursively, leaving unneeded abstract terms local in their abstract layer.

We focus here on the third alternative.

5. An Example for Limited Domain Sharing

A multi-domain algebra needs the knowledge about the domains, specifically about the semantics of the intersecting terms.

We will illustrate the concept with a simple example:

S Domain **S** is of shoe stores, with objects as shoes to be sold, customers, their feet, sales people, business locations, and suppliers.

F Domain **F** is of shoe factories, with shoes being produced, lasts, materials as leather, glue, nails, and thread, suppliers for the material, employees, and production machinery.

In order to create an information system that combines data from both, it is not necessary to merge both the **S** and **F** ontologies completely. Only terms along their connections must be merged, we assume by default that terms do not match. The required knowledge is:

S:`supplier.name`	=	**F**:`factory.name`
S:`shoe.size`	=	**F**:`shoe.size`
S:`shoe.color`	=	`color_table (`**F**`:shoe.color)`

The color table provides the translation between the colors being attached to sales items, such as `pretty pink` and the color designation used in the factory, say, `13XF3`. Sometimes such relationships can be expressed as functions, say, conversions from `cm` to `inches`.

Not included in the knowledge-base, and hence not composable is the term `nail`, which in the store domain **S** is part of the customer's anatomy, and in the factory **F** designates part of the material used to make shoes. Similarly, the `employees` remain distinct, since the data collected for sales people differ from those in the factory.

The attached Figure illustrates the issues.

—

—

The income tax domain **I** will establish other connections between it and the sales and factory domains. A department store, incorporating many sales subdomains, will have more semantic connections, but still avoid needing an unconstrained union of all its ontologies.

We achieve scalability of information systems in this approach by the ontological partitioning [Gruber:94]. We enable composition over the parts by having a knowledge-based algebra. The individuals chartered with defining and maintaining the knowledge need more breadth than those that maintain domain-specific ontologies, but do not need the same depth of knowledge for the shoe supply connection. No knowledge about manufacturing detail is needed, although the factory may provide an abstraction called `quality`.

6. A DKB algebra

Given a formal Domain-Knowledge-Base model (*DKB*) containing matching rules that define sharable terms, the DKB-algebra should contain the following binary operations among domains:

`Operation`	`symbol`	`semantics`
`DKB-Intersection`	$\bigcap(DKB)$	`create a new subset ontology, comprised of sharable entries`
`DKB-Union`	$\bigcup(DKB)$	`create a new joint ontology, labeling all but shared entries with their source`
`DKB-Difference`	$-(DKB)$	`remove entries from an ontology, but shared entries are retained`

Simple negation is avoided, so that no infinite ontologies are created.

Such an algebra can provide a basis for interrogating multiple databases which are sematically disjoint, but where a shared knowledge-base has been established. This process mirrors the approach used in CARNOT, where a knowledge base is used to create *articulation axioms* for joining of data [Collet:91]. However, CARNOT uses the default assumption that everything matches. When CARNOT uses a large and broad CYC knowledge base, many irrelevant retrievals can occur, so that in practice CARNOT system applications limit the depth of search.

An abstract layer created by taking the union ($\bigcup_{(DKB)}$) of several prior intersections ($\bigcap_{(DKB)}$) should not contain so many terms that coherence is hard to achieve. The relative autonomy of the local source terms provides scalability. The layered structure actually abdopts for information structuring the domain management strategy used by the INTERNET distributed naming conventions [Kahn:87].

With the conservative assumptions embedded in the *DKB-model*, the risk is that, because of having insufficiently many matching rules, too little information will be retrieved. By assigning the task of creating matching rules to many expert groups, we expect that high quality operations over data from distinct, but overlapping domains can be created at a reasonable cost. To evolve these systems effectively, feedback loops must exist that permit users to suggest new candidate matching rules, or to modify existing ones. Having small, distributed groups to maintain the partitioned *DKB-model*s will help ensure responsive maintenance of the domain knowledge.

7. CONCLUSION

Information technology is serving us well in specific domains, although we have remained dependent on specialist model designers and programmers for the implementation. Object technology lessens our dependence on specialists by being able to use an infrastructure which aggregates detail into meaningful units.

When the breadth of information system grows beyond coherent domains, further knowledge should be incorporated. To profit from such knowledge we propose a knowledge-based information algebra. The tasks of collecting and maintaining such algebras can be naturally partitioned among specialists and collaboratoring integrators. Integration can proceed at multiple levels of abstraction, avoiding the centralization that hinders progress in data exploitation of data from diverse sources.

Tools are needed to support such development, but to have effective tools a common formal structure is needed. Artificial Intelligence technology has been hard to scale when domains grew large or became diverse. The technology we describecan provide the needed formalism by building on relational algebras, formal management of semantics, and the incorporation of ontological concepts as a foundation for the management of the required knowledge bases.

REFERENCES

[Acton:93]C.H. Acton: "Using the SPICE System to Help Plan and Interpret Space Science Observations"; *Proceedings of the Second International Symposium on Ground Data Systems for Space Missions Operations*, Pasadena, California, November 16-20 (JPL Publication No. 93-5, March 1, 1993).

[Barsalou:91]T. Barsalou, N. Siambela, A. Keller, and G. Wiederhold: "Updating Relational Databases through Object-Based Views"; *ACM SIGMOD Conf. on the Management of Data 91*, Boulder CO, May 1991.

[Collet:91]C. Collet, M. Huhns, and W-M. Shen: "Resource Integrating Using a Large Knowledge Base in CARNOT"; *IEEE Computer*, Vol.24 No.12, Dec.1991.

[Dixon:69]Wilfrid J. Dixon and Frank J. Massey jr.: *Introduction to Statistical Analysis*; McGraw-Hill, 1969.

[Gruber:93]Thomas R. Gruber: "A Translation Approach to Portable Ontology Specifications"; *Knowledge Acquisition*, Vol.5 No. 2, pp.199-220, 1993

[Gruber:94]Thomas R. Gruber and Gregory Olsen: "An Ontology for Engineering Mathematics"; in Jon Doyle, Piero Torasso, and Erik Sandewall, Eds., *Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Gustav Stresemann Institut, Bonn, Germany. Morgan Kaufmann Publishers, Inc., May 1994.

[Haddock:94]G. Haddock and K. Harbison: "From Scenarios to Domain Models: Processes and Representations"; *Proceedings of the Conference on Knowledge-based Artificial Intelligence Systems in Aerospace and Industry*, SPIE, April 1994.

[Humphreys:92]B.L. Humphreys and D.A.B. Lindberg: "The Unified Medical Language Project: A Distributed Experiment in Improving Access to Biomedical Information"; *MEDINFO 92*, North-Holland, 1992, pp.1496–1500.

[Kahn:87]Robert E. Kahn: "Networks for Advanced Computing"; *Scientific American*, Vol 257 No.5; Oct.1987, pp.136-143.

[Lenat:90]D. Lenat, R.V. Guha, et al.: "CYC: towards programs with common sense"; *Communications of the ACM*, Vol.33 No.8, Aug.1990.

[Loomis:87]Mary E.S. Loomis: *The Database Book*; MacMillan, 1987.

[Rose:83]Robert F. Rose: "A 'Data Engine' Using SAS and INQUIRE"; *Journal of Medical Systems*, Vol.7 No.3, 1983, pp.257–266.

[Wiederhold:91]Gio Wiederhold: "The Roles of Artificial Intelligence in Information Systems"; *Journal of Intelligent Information Systems*, Vol.1 No.1, 1992, pp.35–56.

[Wiederhold:92]Gio Wiederhold, Peter Wegner, and Stefano Ceri: "Towards Megaprogramming"; *Comm. ACM*, November 1992, pp.89-99.