Semantic Interoperability in Information Services:
Experiencing with CoopWARE

Avigdor Gal
Department of Management Sciences & Information Systems
Rutgers University
Email: avigal@rci.rutgers.edu

1 Introduction

Obtaining high quality information has become in recent years a challenging task as data should be gathered and filtered from a large, open and frequently changing network of distributed data sources, with blurred semantics, and no central control over the data sources' structure and availability. This technical challenge is highlighted by the advent of the World Wide Web, its current status and its evolving nature. This state of affair creates a need for technologies for building *information services*, capable of providing high quality information obtained from distributed, heterogeneous, and autonomous data sources on an as-needed basis. While the use of information brokers exist for some time now (e.g. ORB in the CORBA model [13], HTTP- Hypertext Transfer Protocol- requests, Java virtual machines embedded in Web browsers to run applets, or remote procedure call protocols), researchers and practitioners alike are coming to realize that any technology for information services should tackle head-on the problem of semantic interoperability, i.e. the capability of an application to exchange data and activate data manipulation functions by utilizing its domain model. Therefore, architecture for information services should first and foremost handle the semantic issues in providing information services. Within this architecture, tools for semantic understanding of heterogeneous, distributed, and autonomously evolving data sources should be developed in order to provide a transparent representation of the domain regardless of the underlying data sources.

In order to provide coherent and current information in a frequently changing environment (e.g. the Web), information services cannot rely on static ontologies as was suggested, for example, in the SIMS project [1]. Nor can they rely on the collection and reconciliation of semantic information at query time (as proposed in the dynamic classificational ontologies model [17] and also utilized in SCOPES [27]). Towards this end, consider the availability problem that is so common to many Web servers, which interferes with the gathering of the required semantic information at time of retrieval. Therefore, information services should take initiatives in adopting their ontologies to the continuously changing semantics of the data sources.

In this work we propose a coordination mechanism to serve as a basis for generic architecture for information services [23], which generates a domain model of the application using a reactive approach. We suggest a mechanism that is based on conceptual modeling techniques, where concepts are being defined and refined within a metadata repository through the use of instantiation, specialization, and attribution. Also, active database techniques are exploited to continuously provide robust mechanisms for maintaining a consistent domain model in rapidly evolving environments. Such a mechanism becomes handy in handling the reconciliation of ontologies among structured, as well as semi-structured, data sources in the support of dynamic integration of autonomous and heterogeneous data sources with possibly evolving and incompatible internal semantics.

CoopWARE (Cooperation With Active relationships Enforcement) [11] is a generic coordinator prototype, built and maintained at the University of Toronto, which was designed according to the aforementioned guidelines. CoopWARE adopts an active approach in designing ontologies as a way of achieving semantic interoperability. As a generic architecture, CoopWARE has been successfully deployed in three areas, namely collaborative reverse engineering [23], wrappers in digital libraries [11], and information services for the Web [20]. The paper introduces CoopWARE's semantic model (Section 2) and its deployment within the information services setting (Section

3 and Section 4).

As a concrete case study, Section 5 demonstrates the CoopWARE approach in the design of a semantic layer on top of organizational Web sites. Web resources made available through such Web sites have more structure in their design than public Web pages since they share common subject matters. Therefore, one can exploit this added structure in creating conceptual models. In this paper we discuss the semantic aspects of building such a domain model, focusing on university Web sites.

2 The semantic model

The semantic model for information services borrows from the area of conceptual modeling [5]. *Conceptual modeling* formally describes aspects of a modeled reality for the purposes of understanding and communication [22], and therefore is compatible with more recent approaches [3], [31]. The process of conceptual modeling results in a *conceptual schema*, represented in some *conceptual model*. Before presenting the conceptual model chosen to support information services, let us first examine its use in this framework more closely. Information services use conceptual schemata as reference models according to which communication is interpreted. Hence, a conceptual schema represents only as much of the modeled reality as needed for a correct execution of the information service that utilizes it. Therefore, the process of conceptual modeling differs from knowledge representation (in the AI sense) where a complete reflection of the modeled reality is required for an unspecified intelligent task, to be performed by a computerized system in the future [4]. Also, the conceptual schema utilized by an information service changes frequently, either due to changes in the modeled reality or due to the need to expand/truncate/modify the schema to adapt to changes in the information service's components. Obviously, these changes cannot be efficiently captured by using data modeling techniques, nor can they be captured by using semantic data modeling [29], which was designated in the '80s as a support tool for designing databases. As such, semantic modeling is too closely related to the physical aspects of design which cannot be considered pertinent, given the heterogeneous environment in which information services operate.

We adopt the semantic model as suggested in Telos [22], which is flexible enough to support dynamic changes in ontologies in a natural way. A conceptual schema $\mathcal{CS} = (V, E)$ is a digraph with a set of vertices V, representing *concepts*, and a set of edges E over V such that $E = \{\langle v_i, c_{ij} : n_{ij}, v_j\rangle | \{v_i, v_j\} \subseteq V\}$.[1] Concepts are terms within the modeled reality which are deemed important for the communication of information services. An edge between two vertices represents a relationship between two concepts in the modeled reality, identified by a *category*, and a *name*, annotated above as $c_{ij} : n_{ij}$, whose role in the model is explained shortly. An edge direction is interpreted as follows: "v_j is a *property* of category c_{ij} with a name n_{ij} that is mapped onto domain v_i." The relationship can be either a model-specific relationship, which represents either a semantic resemblance or a semantic relevance using semantic proximity categorization [18], or can relate to a pre-defined category. In this model, there are two pre-defined categories, namely IN and ISA (the latter represents a semantic relationship using semantic proximity categorization). The former annotates an instantiation relationship, while the latter annotates a specialization relationship.

Instantiation and specialization are thoroughly discussed in the literature (e.g. [16]). Instead of debating their philosophical role in a modeled reality, we present next their technical implication on the conceptual schema. Let $\{v_i, v_{j1}, v_{j2}, ..., v_{jq}\} \subseteq V$ be $q + 1$ concepts of interest in the modeled reality, such that $\forall v_{jl} \in V$, $\langle v_i, \mathsf{IN}: n_{ijl}, v_{jl}\rangle$. Therefore, $\forall \langle v_i, c_{ik} : n_{ik}, v_k\rangle \in E, \exists \langle v_{jl}, c_{jlp} : n_{jlp}, v_p\rangle \in E | n_{jlp} = c_{ik} \wedge \langle v_k, \mathsf{IN}: n_{kp}, v_p\rangle \in E$. In simple terms, properties can also be instantiated (where a label at one level is instantiated into a category in a lower level) and the scope of a relationship is strictly enforced over the instantiation lattice. For example, consider a university ontology, which consists of Jane Doe, a faculty member. The concept Faculty has a label GrantsAwarded, which is instantiated into a category GrantsAwarded, and its values are strictly constrained to the GrantsAwarded domain, as defined within the Faculty concept. It is worth noting that while Telos allows any number of instantiation levels, the common approach towards attribution in object-based models (e.g. CODM in [17]) utilizes a strict two level approach, where an instance of an attribute (a category in our terminology) is a value that cannot be further instantiated.

The ISA category takes on the common interpretation of generalization/specialization, where a specialized concept inherits all categories from its predecessors, and the scope of the category is strictly enforced.

[1] The Telos notation refers to propositions, where a proposition p is a triplet $\langle from, label, to\rangle$. Interpreting a set of propositions as a digraph is straightforward.

Formally speaking, let $\{v_i, v_{j1}, v_{j2}, ..., v_{jq}\} \subseteq V$ be $q+1$ concepts of interest in the modeled reality, such that $\forall v_{jl} \in V$, $\langle v_i$,ISA: $n_{ijl}, v_{jl}\rangle$. Therefore, $\forall\langle v_i, c_{ik} : n_{ik}, v_k\rangle \in E, \exists\langle v_{jl}, c_{jlp} : n_{jlp}, v_p\rangle \in E | c_{jlp} = c_{ik} \wedge (\exists v_r | \{\langle v_k, \text{IN}: n_{kr}, v_r\rangle, \langle v_p, \text{IN}: n_{pr}, v_r\rangle\} \subseteq E)$.

3 Conceptual schemata for information services

Having introduced the semantic model, we are now ready to demonstrate its deployment within the context of information services. A conceptual schema for an information service can be roughly partitioned into two parts, namely the domain ontology [14] and the system references. While the former is utilized in order to arrange the information in a fashion which is comprehensible to service's users, the latter provides the needed information on how to retrieve the data from the data sources and how to process it into usable information. Elements from the two parts of the schema interact through relationships that relate domain concepts with relevant information on the one hand, and allow the classification of information into correlating domain concepts, on the other hand.

As an example, consider a university Web site as a data source for an information service. The university's domain ontology consists of a concept Student. This concept is associated with one (or more) WebArtifact within a university Web site, where a WebArtifact is a concept in a conceptual schema, which is part of the system reference. Consider next the following two specific examples:

1. The Student concept at the University of Toronto is associated with a WebArtifact of which URL is http://www.toronto.edu/students.html. Therefore, there exists an instantiation of WebArtifact that instantiates the property attribute:URL into URL:http://www.toronto.edu/students.html.
2. The Student John Doe maintains a WebArtifact at the University of Toronto Web site with the URL http://www.cs.toronto.edu/~jdoe. In this case, there is an instantiation of the Student concept, which instantiates the property attribute:Name into Name:JohnDoe and is associated with an instance of WebArtifact with the property URL:http://www.cs.toronto.edu/~jdoe.

These two examples demonstrate a modeling capability that cannot be handled directly by using data modeling tools such as Object-Oriented Analysis [6], [9]. Particularly, our model allows the flexibility of having inter-level links, where both a "class" and an "instance" (in OO terms) may refer to an "instance" through relationships, other than IN (similar to Java reflection, yet with an infinite number of levels).

Next, we extend the discussion to handle heterogeneity. As with many other research projects (e.g. [21], [17]), we use a global conceptual schema for resolving heterogeneity issues. Specialization becomes particularly handy in consolidating domain ontologies with a semantic proximity (loosely generalizing from [33][2]). For example, consider the university domain, and assume now that the information service handles the Web sites of several universities, including the University of Toronto and Rutgers University. While the University of Toronto uses a single term (a Student) to identify those who wish to become students as well as those who are already enrolled, Rutgers University differentiates a ProspectiveStudent from a CurrentStudent. A global conceptual schema includes both ProspectiveStudent and CurrentStudent as specialized concepts of the Student concept.

The level of complexity of the homogenization process can be greatly reduced by using *fusion* to consolidate related terms, with slight semantic shifts, in several data sources.[3] The proposed conceptual model allows different schemata and instance data from autonomous, heterogeneous data sources to be fused by using three possible techniques, or their hybrids:

Multiple inheritance: Concepts can be fused prior to instance data gathering. This requires prior knowledge of the ways instance data will be fused. For example, if a faculty member can be associated both with a department (making her a faculty member) and with a graduate school (making her a supervisor), a common faculty-supervisor subconcept (using the ISA relationship) is defined before it is instantiated. This technique is useful for pre-defined, long-term relationships among concepts, which makes it the most wide-spread method in a database context.

Multiple instantiation: A single concept can be declared as an instance of the concepts faculty

[2]The types of semantic proximity, as presented in [33] do not completely overlap with our definition of consolidation. For example, we do not differentiate semantic resemblance from semantic relevance. Also, the various aspects of fusion (see below) cannot be captured by any of the classes of semantic proximity.

[3]The fusion operation was also introduced in the LOREL system [28] in a more restricted way. Also, fusion was dealt with, using different terminology in [27].

and supervisor. No prior knowledge of this combination is required in the declaration of the schema, so it provides the designer with the freedom of defining case-based relationships.

Instance fusion: Here faculty and supervisor concepts can have independent instances, which can be fused at the instance level by multiple inheritance of instances, leaving the concepts unfused. Again, no prior knowledge of this combination is required at schema-declaration time.

4 Schema design for information services: tracking a moving target

Schema design for information services is inherently different from traditional database systems design. In the latter, the design (or redesign) process is manually performed by the DBA, taking into account organizational considerations, and its output is handed down to the DBMS in some DDL. Contrary to that, the design of conceptual schemata for information services has the following properties:

Bottom-up design: The design of each of the existing data sources proceeds the conceptual schema design. Therefore, the conceptual schema consists of a component of homogenizing heterogeneous ontologies (e.g. [17], [7], and [30]), in addition to the classical role of a designer, which selects concepts to be covered by the database.

Continuous monitoring: The design of conceptual schemata for information services consists of a monitoring component, in which changes in the design of the data sources are detected and propagated to the conceptual schema for correct execution. Some work in this direction was performed in the framework of federated databases [32], yet the absolute majority of models assume either a one-time (or infrequent) reconciliation process, or a cooperative nature of the data sources [15]. As an example to the need in continuous monitoring, consider a set of heterogeneous databases, each accessible through a Web interface. In most cases, the local schemata are not available using the interface. Moreover, the local databases are not bound to inform the users of schema changes. Therefore, the interface should be monitored continuously to identify local changes and incorporate them in the global schema. It is worth noting that continuous monitoring carries with it an overhead. However, it is preferred in situations where the changes are frequent (and thus long delays are expected while establishing the incremental changes to the ontology at query time) and is unavoidable where availability is not guaranteed (see below).

Rapid modifications: Changes to the underlying data sources are rapid. These changes include not only changes to the "data," but also to the "metadata" (in database terms) [10]. Therefore, the design of conceptual schemata for information services should be efficient in order to scale-up.

Availability: Environments such as the Web are unstable, due to each information source's autonomous decision of availability. Therefore, it is possible that at the time of query, the required schema will not be available at the information service's disposal. Towards this end, consider a network of Web pages which reside on several machines, some of which are possibly unavailable at any point in time. If an ontology is not built using continuous monitoring, it cannot be completed at query time, as some links may be missing.

Observing these four properties, it becomes evident that alternative methods are necessary in order to support the design of conceptual schemata for information services. In particular, none of the methods suggested for distributed or heterogeneous databases handle the combination of all four properties. The research agenda of distributed databases assumes a central control over their design, research efforts in heterogeneous databases disregard the last three properties, and dynamic schema extractors (e.g. [27] and [17]) although capable of handling semi-structured data assume an incremental building process of ontologies combined with query-time domain model analysis and therefore do not support the second property and are less effective in the presence of the fourth property. Towards this end, we suggest to employ an approach that consists of two components:

Coordinator: a set of tools and mechanisms that enable active mediation [35] among data sources. The coordinator enables the execution of information services in a distributed heterogeneous environment while supporting an active self-analysis of the continuously changing data sources. In this sense, we extend the use of a co-

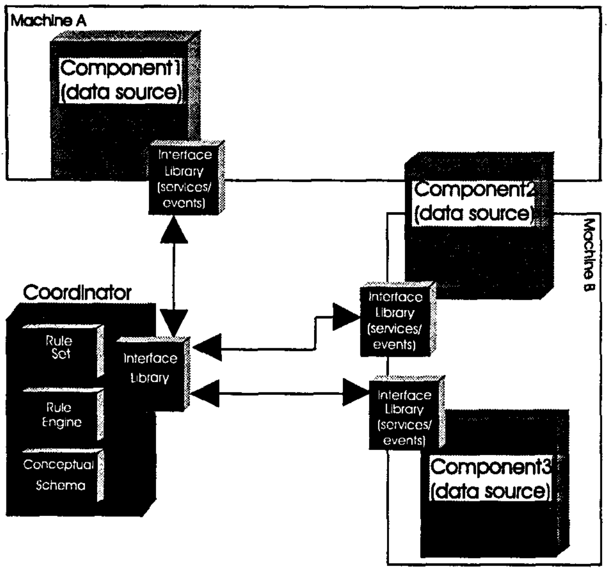

Figure 1: The CoopWARE architecture

ordinator for dynamic resolution of semantic heterogeneity, as initially suggested in [25], to perform the resolution process as soon as the update occurs, rather than in query time. It is worth noting that the coordinator, as described in [25] and [26] uses a learning process which increases the performance of a single query as the number of queries increase.

Information repository: the part of the environment that manages the conceptual schema. Its main function is to serve as an information broker [18] among the various components of the architecture and to provide a flexible tool for abstracting and modifying methods of operation.

CoopWARE implements a generic integration architecture, which supports the four properties mentioned above through asynchronous communication. Figure 1 demonstrates the general structure of the information services architecture of CoopWARE. It consists of several conceptual *components* (a conceptual component can extend beyond the boundaries of a single machine) that communicate through a coordinator. Each component interacts with structured, unstructured and semi-structured data sources, and has an interface which defines a set of *services* that can be executed by the component and a set of *events*: compact reliable occurrences that enable the flow of information regarding the state of the domain model. A coordinator contains a conceptual schema, a rule mechanism (referred to as "Rule Set" and "Rule Engine" in Figure 1), and an interface. A detailed account of CoopWARE design can be found in [12]. An example of a design using CoopWARE follows.

5 Information services for the Web: building and maintaining conceptual schemata

In this section we share our experiences in designing information services using CoopWARE. We focus on integrating existing information sources, available via the Web, in the delivery of information services. The information sources may include databases, formatted or plain ASCII files, and other computer-based data such as Java applets, embedded CGI scripts (with DOM- Document Object Model- [8] as a standard way to interact with documents, including naming elements and associating event handlers with elements), stream media, 3D graphics, and clickable maps. These sources may be multiple, distributed and heterogeneous (as long as they share, at least to some extent, the same context [24]). They may also contain legacy data in that their original designers are long-gone and their semantics are only partially understood.

Information services in this setting use three types of information, namely Web artifacts, domain concepts, and change propagation rules for maintaining semantic consistency as the site changes. Web artifacts are purely syntactic entities found in HTML or XML (eXtensible Markup Language) [36] files, and HTTP or FTP requests. The repository representations of these artifacts are automatically generated and regenerated by a Web extraction tool. Such a tool is syntactically based, i.e. it does not reflect the subject matter (domain) the Web documents are concerned with.[4] Domain concepts reflect the ontology of the application's domain. A concept is possibly associated with other concepts through domain attributes that reflect the relationships among domain concepts. Also, a domain concept has one or more associated Web artifacts that are bound to the concept by an attribute WebArtifact. Domain concepts are initially generated by examining the hyperlink graph topology and the content of each Web site. Each artifact in a site initially generates a corresponding domain concept, where its name is taken from either the Web document title or the link label. The hyperlinks in the document (and tagging in the case of XML)

[4] While XML has the capability of tagging information based on specific application's ontology, these tags are not readily visible to the user.

constitute the properties of the concept.

Next we describe the role of change propagation rules in establishing semantic consistency. CoopWARE uses three separate components for the active maintenance of the semantic schema, namely WebMonitor, Designer, and ConceptEditor. WebMonitor monitors a set of URLs given to it through the services AddToScope and RemoveFromScope. It detects and informs the coordinator of an insertion, modification, removal or relocation of Web artifacts within its scope. The detection process is performed either by using a periodic polling or by having a proxy that detects these changes automatically, e.g. BackWeb [2]. Generally speaking, the former is more costly than the latter in terms of network traffic, yet when the scope of the search is localized (as is the case with an intranet) the benefits of the latter are less obvious. Also, push technologies, as the one used in BackWeb, necessitates a proxy at the server's end to avoid network congestion. This is not always feasible, as we do no have control over the Web resources. The designer is embedded as another component of the architecture, which offers the services of both suggesting a domain concept and verifying an existing domain concept. The communication with the designer is asynchronous and can be handled either via email, or through the use of a worklist, where the designer is prompted whenever such a request is pending. The ConceptEditor controls the domain model as stored in the repository. This can be done by using a variety of techniques, e.g. MIRROR [19] and SCOPES' coordinator [27].

The detection of changes within the application's scope is performed by using events. In this context, we support four types of events, namely *artifact insertion*, *artifact modification*, *artifact removal*, and *artifact relocation*. Each event transfers parameters to be used by CoopWARE and the services of other components. The consequences of event detection are captured in rules. A rule is a programming mechanism that is constructed of three segments, namely an event, a condition, and an action (ECA) [34]. The semantics of a rule are as follows: when an event ev occurs, conditions $co_1, co_2, ..., co_n$ are independently evaluated, where condition co_i $(1 \leq i \leq n)$ is part of a rule r_i such that $ev_i = ev$. If co_i $(1 \leq i \leq n)$ evaluates to true, then ac_i is activated. For example, whenever an artifact insertion event occurs, the artifact is added to the repository (using the AddArtifact service request) and facts about the artifact's state are recorded in the repository (using the RecordArtifactState service request). These facts include the document's HTML title and the links contained in the file (both their labels and target URLs). When a new artifact is inserted, the designer is introduced with a provisional domain concept corresponding to the HTML title, the file name, or the link label using the SuggestDomainModel service request. This concept is tentatively added to the semantic model (using the AddConcept service request) and is possibly removed at a later time, should the designer decide not to accept the system's suggestion. Thus, the following rule is utilized by the information service:

Event:	ArtifactInsertion(URL)
Condition:	InScope(URL)
Action:	AddArtifact(URL)
	State:=ExtractArtifactState(URL)
	RecordArtifactState(State)
	SuggestDomainConcept(URL, URL.title)
	AddConcept(URL.title)
	Associate(URL, URL.Title)

The SuggestDomainConcept uses the data that was gathered through RecordArtifactState in offering a tentative domain concept to the designer (which is related to the page's title in this case). The execution of the rule is conditioned upon the Web artifact being part of the scope. This mechanism prevents an exponential processing of Web artifacts, which might stem from undistinguished addition of artifacts to the information base due to hyperlinks.

6 Conclusion

The paper demonstrated our approach towards the design and implementation of information services, emphasizing the need for a reactive approach towards semantic interoperability to ensure high-quality output from distributed, heterogeneous, and autonomous data sources. In particular, we have suggested a coordination mechanism that uses active database techniques as a basis for generic architecture for information services. A prototype has been implemented and deployed in several areas, including Web-based information services.

A continuous shift in industrial attention, as exemplified in XML, involves the design of standards for semantic interoperability. These new tools should be evaluated in further research more closely from an academic point of view. Another issue deserving attention involves testing the capabilities of a computerized tool to suggest an accurate ontology without human intervention. As has been demonstrated in other areas before, the design effort would be all for naught if the designer would be required to intervene on a large scale. Towards this end, the use of pre-defined ontologies, as well as AI techniques (e.g. neural networks) should be deployed to evaluate the accuracy of tentative concepts and their relationships with other terms within the ontology.

Acknowledgment

The CoopWARE model and its deployment in Web based information services is done in collaboration with John Mylopoulos and Scott Kerr.

References

[1] Y. Arens, C.A. Knoblock, and W. Shen. Query reformulation for dynamic information integration. In G. Wiederhold, editor, *Intelligent Integration of Information*, pages 11–42. Kluwer Academic Publishers, 1996.

[2] The BackWeb home page. http://www.backweb.com.

[3] D. Beech. Data semantics on the information superhighway. In R. Meersman and L. Mark, editors, *Database Application Semantics*. Chapman and Hall, 1997.

[4] A. Borgida. Knowledge representation, semantic data modelling: What's the difference? In *Proceedings of the 9th International Conference on Entity-Relationship Approach (ER'90)*, pages 1–2, Lausanne, Switzerland, 1990.

[5] M. Brodie, J. Mylopoulos, and J. Schmidt, editors. *On Conceptual Modelling: Perspectives from Artificial Intelligence and Programming Languages*. Springer Verlag, 1984.

[6] P. Coad and E. Yourdon. *Object Oriented Analysis*. Prentice Hall, Englewood Cliffs, 2nd edition, 1991.

[7] R. Daniel, C. Lagoze, and S. Payette. A metadata architecture for digital libraries. In *Proceedings of the Forum on Research and Technology Advances in Digital Libraries (ADL'98)*, pages 276–288, Santa Barbara, CA, 1998.

[8] Document Object Management. http://www.w3.org/DOM.

[9] D. Dori. Unifying system structure and behaviour through object-process analysis. *Journal of Object-Oriented Analysis*, pages 66–73, July-August 1996.

[10] A. Gal. Handling constantly changing metadata. In *the Proceedings of the Second IEEE Metadata Conference (http://computer.org/conferen/proceed/meta97/)*, September 1997.

[11] A. Gal and J. Mylopoulos. The CoopWARE demo: wrapping up a legacy system. http://www.cs.toronto.edu/~coopware, 1997.

[12] A. Gal and J. Mylopoulos. Supporting distributed autonomous information services using coordination. *International Journal of Cooperative Information Systems*, 1998. Accepted for publication. Available upon request from avigal@rci.rutgers.edu.

[13] The Object Management Group. The common object request broker: Architecture and specification. Technical Report 91.12.1 Rev 1.1, Object Management Group, December 1991.

[14] N. Guarino. Formal ontology and information systems. In *Proceedings of the 1st International Conference on Formal Ontology in Information Systems (FOIS'98)*, pages 3–15, 1998.

[15] D. Heimbigner and D. McLeod. A federated architecture for information management. *ACM Transactions on Office Information Systems*, 3(3):253–278, July 1985.

[16] R. Hull and R. King. Semantic database modeling: Survey, application and research issues. *ACM Computing Surveys*, 19(3):201–260, Sep 1987.

[17] J. Kahng and D. McLeod. Dynamic classification ontologies for discovery in cooperative federated databases. In *Proceedings of the First IFCIS International Conference on Cooperative Information Systems (CoopIS'96)*, pages 26–35, Brussels, Belgium, June 1996.

[18] V. Kashyap and A. Sheth. Semantics based information brokering. In *Proceedings of the 3rd International Conference on Information and Knowledge Systems*, pages 363–370, 1994.

[19] S. Kerr. Data integration and change management using active metamodels. Master's thesis, University of Toronto, 1998.

[20] S. Kerr, A. Gal, and J. Mylopoulos. Information services for the web: Building and maintaining domain models. In *Proceedings of the Third IFCIS Internationla Conference on Cooperative Information Systems (CoopIS'98)*, pages 4–13, NYC, NY, August 1998.

[21] A.Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, 5:121–143, 1995.

[22] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4), October 1990.

[23] J. Mylopoulos, A. Gal, K. Kontogiannis, and M. Stanley. A generic integration architecture for cooperative information systems. In *Proc. CoopIS'96*, Brussels, Belgium, June 1996.

[24] J. Mylopoulos and R. Motschnig-Pitrik. Partitioning information bases with contexts. In S. Laufmann, S. Spaccapietra, and T. Yokoi, editors, *Proceedings of the Third International Conference on Cooperative Information Systems (CoopIS-95)*, pages 44–54, Vienna, Austria, May 1995.

[25] A.M. Ouksel. Semantic mechanisms for cooperation in heteregeneous database systems. In *Proceedings of 1992 International IEEE Conference on Man and Cybernetics*, October 1992.

[26] A.M. Ouksel and I. Ahmed. Coordinating knowledge elicitation to support context construction in cooperative information systems. In *Proceedings of First IFCIS International Conference on Cooperative Information Systems (CoopIS'96)*, Brussels, Belgium, June 1996.

[27] A.M. Ouksel and C.F. Naiman. Coordinating context building in heterogeneous information systems. *Journal of Intelligent Information Systems (JIIS)*, 3(2):151–183, April 1994.

[28] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman. Medmaker: A mediation system based on declarative specifications. In *Proceedings of the IEEE International Conference on Data Engineering*, pages 132–141, New Orleans, February 1996.

[29] J. Peckham and F.J. Maryanski. Semantic data models. *ACM Computing Surveys*, 20(3):153–189, 1988.

[30] K. Shah and A. Sheth. Logical information modeling of web-accessible heterogeneous digital assets. In *Proceedings of the Forum on Research and Technology Advances in Digital Libraries (ADL'98)*, pages 266–275, Santa Barbara, CA, 1998.

[31] A. Sheth. Data semantics: What, where, and how? In R. Meersman and L. Mark, editors, *Database Application Semantics*, pages 601–610. Chapman and Hall, 1997.

[32] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[33] A.P. Sheth and V. Kashyap. So far (schematically) yet so near (semantically). In D.K. Hsiao, E.J. Neuhold, and R. Sacks-Davis, editors, *Proceedings of the IFIP WG 2.6 Database Semantics Conference on Interoperable Database Systems*, Lorne, Victoria, Australia, November 1990. North-Holland.

[34] J. Widom and S. Ceri, editors. *Active Database Systems: Triggers and Rules for Advanced Database Processing.* Morgan Kaufmann, San Francisco, CA, 1996.

[35] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.

[36] eXtensible Markup Language at the W3 Consortium. http://www.w3.org/XML.