AFRL-IF-RS-TR-2007-147
Final Technical Report
June 2007

MEDIATION, ALIGNMENT, AND INFORMATION SERVICES FOR SEMANTIC INTEROPERABILITY (MAISSI): A Trade Study

BAE Systems

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Air Force Research Laboratory Rome Research Site Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-IF-RS-TR-2007-147 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/ /s/

GENNADY STASKEVICH
Work Unit Manager

JAMES W. CUSACK
Chief, Information Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JUN 2007	Final	Sep 06 – May 07

4. TITLE AND SUBTITLE

MEDIATION, ALIGNMENT, AND INFORMATION SERVICES FOR SEMANTIC INTEROPERABILITY (MAISSI): A TRADE STUDY

5a. CONTRACT NUMBER
FA8750-06-C-0207

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
63789F

6. AUTHOR(S)

5d. PROJECT NUMBER
SEMI

5e. TASK NUMBER
06

5f. WORK UNIT NUMBER
01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
BAE Systems, Advanced Information Technologies, Inc.
6 New England Executive Park
Burlington, MA 01803

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFSE
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2007-147

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 07-274

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Today's emphasis on joint and combined military operations and the need to maximize the effectiveness of these operations has led to the need to better support interoperability among the various constituent communities. While the emerging net-centric infrastructure allows these communities and their systems to communicate, this dialog is often constrained by the lack of common semantic points of reference. For example, legacy databases often have custom schemas that represent the same information in disparate ways. Semantic Interoperability (SI) encompasses a broad range of technologies such as data mediation and schema matching, ontology alignment, and context representation that attempt to enable systems to understand each others' semantics with minimal modification of the legacy systems.

15. SUBJECT TERMS
Net-centric, Semantics, Interoperability, Information Transformation, Information Management, Study

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Gennady R. Staskevich
U	U	U	UL	80	19b. TELEPHONE NUMBER (*Include area code*) 315-330-4889

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

List of Figures

Section **Page**

1 EXECUTIVE SUMMARY

Today's emphasis on joint and combined military operations and the need to maximize the effectiveness of these operations has led to the need to better support interoperability among the various constituent communities. While the emerging net-centric infrastructure allows these communities and their systems to communicate, this dialog is often constrained by the lack of common semantic points of reference. For example, legacy databases often have custom schemas that represent the same information in disparate ways. Semantic interoperability (SI) encompasses a broad range of technologies such as data mediation and schema matching, ontology alignment, and context representation that attempt to enable systems to understand each others' semantics with minimal modification of the legacy systems.

BAE Systems, Advanced Information Technologies (BAE-AIT) performed a trade study to assess the current state of the art of semantic interoperability technologies to meet the Air Force's need for a scalable and interoperable information sharing environment. For purposes of this study we decomposed the SI space into two areas, *semantic alignment* and *information services architecture*. Semantic alignment addresses how information is represented and how it is transformed from one representation to another. Information services architecture is concerned with how semantic alignment components can be combined and incorporated into a scalable architecture that easily accommodates legacy components and systems.

A recent trade study by the Cross-Domain Semantic Interoperability (CDSI) Working Group [5] on Data Interoperability, states that "Current technologies provide no viable solution for sharing data across the many domains of large enterprises." Although we agree with the general premise that semantic integration for large scale, realistic ontologies is still immature, we believe that current technologies have made significant progress in addressing key point challenge areas, but they have not yet solved the problem of understanding semantics and exchanging knowledge and information across systems in any automated fashion. We also believe, however, that with proper emphasis of research towards the solution of real operational problems and through the vetting of ideas with the user community, many of the existing technologies will grow to become key enablers in achieving integration.

For this study we identified relevant SI technologies and, for each technology, evaluated the technical strengths and weaknesses, the maturity (e.g., research prototype to deployed component) and current level of vendor or contractor support, the relevance to the Air Force mission, the performance and potential scalability to large numbers of users and systems, and the compatibility with existing and emerging open standards. We also identified technology gaps in the current tools and proposed research areas to address those gaps.

2 INTRODUCTION

2.1 TECHNICAL DISCUSSION

"…[We must] leverage information technology and innovative network-centric concepts of operations to develop increasingly capable joint forces. Our ability to leverage the power of information and networks will be key to our success…"

--Former Deputy Secretary of Defense Paul Wolfowitz

The Air Force (AF) is increasingly under pressure to not only perform missions on its own, but also coordinate its activities with other commands and centers of excellence to participate in missions of global scale. A key challenge in establishing these operations is to exchange data and information across multiple large-scale systems at the application level to enable effective collaboration among multiple Communities of Interest (COIs).

In these situations, the AF must enable vertical and cross-service interoperability with joint-level strategy tools and data sources at national-level commands such as STRATCOM and JFCOM, as well as those of the various regional Combatant Commands and their non-air components. Each command and each echelon has a unique mission, and an established but evolving technology base of databases, schemas, tools, and processes designed to facilitate execution of that mission. Enabling applications to interoperate on a global scale requires dynamic transformations of the data to the specifications of the target systems. In addition, new breakthroughs in Net-Centric Operations have motivated the development of DoD's Global Information Grid (GIG), which provides a vision for interconnecting the information systems across all elements of the national power (Diplomatic, Military, Economic, Civilian support, etc). In this environment, data must be converted not only to different syntactic representations, but also to the semantic models of each domain application.

Semantic Interoperability is the set of technologies and tools that address these challenges and offers a robust architecture for dynamic data transformation across multiple domain problems.

The next section, Section 2.2, describes the technical challenges inherent in achieving SI among disparate legacy systems, and Section 3 exemplifies these challenges using an example scenario for campaign and strategy interoperability. Sections 4 and 5 describe the technologies in detail, and Section 6 lists some key programs that deal with the problem of system integration and discusses the approaches they have taken to address them. Our key observations and recommendation are summarized at the beginning of the report, in Section 2.3, and described in more detailed in the follow-up technical sections. Finally, Section 7 concludes with a summary of conclusions from the trade study.

2.2 TECHNICAL CHALLENGES

Solving the Semantic Interoperability problem requires advancements in two closely related challenge areas, which in turn can be decomposed into further sub-areas. Our proposed trade study will explore tools and technologies for each one of these following challenge areas.

- How to **represent and mediate** information relevant to Air Force missions to enhance net-centric operations and collaboration among different COIs, and
- How to enable **interoperability** among legacy systems and data sources that have been developed by multiple commands and echelons often using completely different technologies and protocols.

We classify these challenges under the two overarching terms of *Semantic Alignment* and *Information Services Architecture*.

Semantic Alignment

Semantic Alignment refers to the combined problems of *information representation* and *mediation* (transformation). If multiple systems are to successfully exchange data, they must first agree on the way they represent information. Developing a globally agreed common *information representation*, however, is very difficult to achieve and almost impossible to maintain. Information Systems support specific domain problems, and the objects they represent are widely different from each other. In many cases there are reasons to keep representations separate (for example, precise electronic measurements from tracking systems shouldn't be combined with imagery data from airborne assets), and there will always be the need to communicate with a non-conforming system (import weather modeling data to AF planning systems). Compounding this problem is the need to deconflict information from multiple source databases, or consolidate attributes of real-world entities that are modeled across multiple external sources. The problem becomes even harder as the source databases evolve over time, and databases get consolidated to capture information for new domain problems. In most instances precise matching between tuples is not possible, yielding the need to reason under schema and semantic uncertainty.

The second alignment challenge is how to efficiently mediate information among multiple cross-COI systems and services. Technologies and standards for mediation at the lowest-level of richness, such as Extract Transform and Load (database mediation) and XSLT (XML mediation) are relatively mature and widely used. However, to enable information on a global scale to flow seamlessly, the more difficult challenges of 1) automated transformation from elementary formats to richer ontology-based formats, and 2) mediation between different domain ontologies and ontology representations must be addressed. Manually encoded ad hoc translators are a first step, but must eventually be replaced by increasingly automated solutions that can infer the appropriate ontology mappings and translation rules. Furthermore, techniques to represent, learn, and apply *contextual information* will be needed to clarify semantics of like information elements within different applications and COIs. This contextual information is also needed to support *semantic enrichment,* in which the information representation and mediation dynamically evolve to support changing information needs of COIs.

Information Services Architecture

Technologies for semantic alignment alone are simply elements of the SI solution. Putting them together to achieve SI in practice will require a robust, layered architecture of *information services*, built on industry standards and enterprise technologies. The Air Force often coordinates complex operations in a collaborative environment with other Commands and Echelons, which requires an infrastructure to support dynamic coupling of software applications to register their

services on the network, discover other applications to interface with, and exchange information via a well-defined and agreed protocol.

In addition, as the number of tools and users grows, the Information Infrastructure must be scalable and extensible to eliminate redundant data storage and transfer, and to avoid proliferation of specialized native data models and peer-to-peer interfaces. The Infrastructure must be able to support a wide spectrum of information richness, ranging from elementary database content to self-describing text formats such as XML to hierarchically organized domain ontologies. Applications that operate in this environment must authenticate and authorize users via well established and trusted access frameworks, disseminate information on a per-need basis and provide services for federated access. These services must be implemented with open industry standards and utilities that are mature, widely accepted by the technical community and maintainable. Finally, we must incorporate legacy systems into the Information Infrastructure by bringing to bear the semantic alignment tools discussed above to automate, or partially automate, the extraction of semantic information from these systems.

2.3 KEY FINDINGS AND RECOMMENDATIONS

In our trade study we examined a wide range of technologies in Semantic Alignment and in Information Services. In addition, we reviewed major programs that attempt to either integrate systems together or try to address the need of a common interoperability infrastructure (both framework and data modeling). Our key findings are that:

- The performance of the tools in the Semantic Alignment category varies widely based on the domain problem and the instance of the evaluation ontologies. In the Ontology Alignment Evaluation Initiative, conducted in 2005 against a bibliographic and anatomic ontologies, results ranged from:
    - Best : 0.91 precision, 0.89 recall, to
    - Worst: 0.08 precision, 0.18 recall
- We did not find evidence of any comprehensive effort to evaluate Semantic Alignment technologies in the context of military domains and DoD problems.
- Information Services technologies require heavy technical knowledge for configuration and composition of component services. Most of the current major SOA related programs expose applications via web services, but integrating these services into a functional system requires heavy involvement by humans to interpret the ICDs, consult with the modelers to understand the semantics of the data and learn from the system developers the use of the data by the application.
- Standard languages do not produce interoperable data. Standard languages have substantial value in expressing concepts in a common representation, but this is not sufficient for data interoperability. Even though OWL, the leading ontology specification language today, provides the ability to import ontologies and express mappings between individual entities across ontologies, this does not solve the pervasive challenge, which is to understand the semantics or the meaning of the individual ontologies.

- Data Interoperability has multiple layers and they are all critical to systems integration. At the bottom of the stack are the bits and bytes and the protocols between hardware platforms. The next level up captures the models and representations of the data, followed by expressive ontologies that describe relationships and dependencies. Finally at the top of the stack is the reasoning and inferencing of new concept and relationships, proof of the results and trust that the machine interpretation is correct.

Our analysis has shown that Systems Integration is still a manual process. There are tools available today to automate many of the steps, such as generation of interface proxies for SOA frameworks, deployment facilities, IDEs (Integrated Development Environments), debuggers, reliable messaging systems, standards, etc, but the need to integrate data is still pervasive and many of the efforts require heavy manual involvement

To advance the state of the art in Semantic Integration, we recommend the following list of research efforts to the government:

- Develop sound theoretical foundations for schema and ontology matching. This effort will guide the research of new techniques in semantic alignment and provide a unique methodology for developing mathematically rigorous automated tools
- Conduct better evaluations and comparisons of matching tools to identify the maximum performance that is theoretically attainable. Without a rigorous evaluation approach, it will be impossible to quantify the capabilities of the technologies and assess the progress in the field of semantic integration.
- Extend semantic discovery by exploiting context and rich constraint information. Several schema matching tools use edit distance and structural similarities to compare and match data models. In addition to these techniques, we propose to explore the expressive information that is captured by ontologies and the context of the data source to do the mapping
- Explore user-assisted semantic alignment. Fully automated matching for complex, realistic ontologies is many years in the future. In the meantime, humans are still better equipped than machines to understand semantics and meaning. The purpose of this effort will be to explore the synergy of automated and user-assisted semantic alignment and find the best way to involve the humans in the integration process.
- Exploit the use of modeling technologies for expressing doctrinal procedures. Instead of letting ontologies grow organically and then try to integrate them, an alternate approach will be to achieve integration through modeling. Express all the aspects of the enterprise (data, processes, business logic, specifications, etc.) in a modeling framework, such as the OMG Meta-Object Facility (MOF), and then use the transformations intrinsic to MOF to convert data and processes between the various components of the enterprise. A critical key to success of this approach is the adoption by the various participants, so a product first step will be to assess the viability of this approach and then explore how easily will be to describe the complexities of each component in a common modeling representation.
- Conduct competition of semantic technologies. Finally, in order to build an environment that naturally selects the most appropriate technologies and stimulates a

healthy competition between the researches we suggest the creation of a Semantic Alignment challenge effort. This program will be similar to the DARPA Challenge and the IC sponsored Knowledge Discovery and Dissemination challenge (KDD), in which the government selects appropriate programs of interest and then invites contractors to compete with their best technologies to solve the needs of the program. This effort will focus the research activities on real operational problems.

In the remainder of this paper we describe in detail the technical characteristics of represented technologies. As we will demonstrate, the need for semantic integration is paramount to the government and the technical challenges have not yet been solved with the current tools. Several approaches have been tried in the past, many of them using variation of upper ontologies, others are applying machine learning tools to discover the semantics of the component sources, yet others are looking at how to enable COIs to grow their ontologies over lateral, associated domains. We believe that the combination of these approaches and the development of further research programs, along the lines of the ones we identified in our report, will provide the means to advance the state of the art in this problem and yield accurate technologies to facilitate the integration of heterogeneous systems.

3 DEMONSTRATION SCENARIO

We will illustrate the use of the Semantic Interoperability technologies via a scenario that involves the intelligence cell of the Air Operations Center as they are following the steps of the Predictive Battlespace Awareness process. PBA consists of four major steps:

1. Intelligence Preparation of the Battlespace. This is the step in which raw information from a variety of source is turned into actionable intelligence report.
2. Target Systems Analysis focuses on the identification of the enemy nodes that if prosecuted will achieve the maximum intended effect.
3. ISR & Master Air Attack Planning is when the planning cells of the CAOC use operational C2 systems to create and compare COAs and choose the most appropriate one that addresses the Commander's Intent
4. ISR Employment & Kill Chain. During plan execution, this steps matches weapons to targets, authorizes the prosecution of targets and allocates the ISR assets to collect intelligence reports in order to asses the status of the mission.

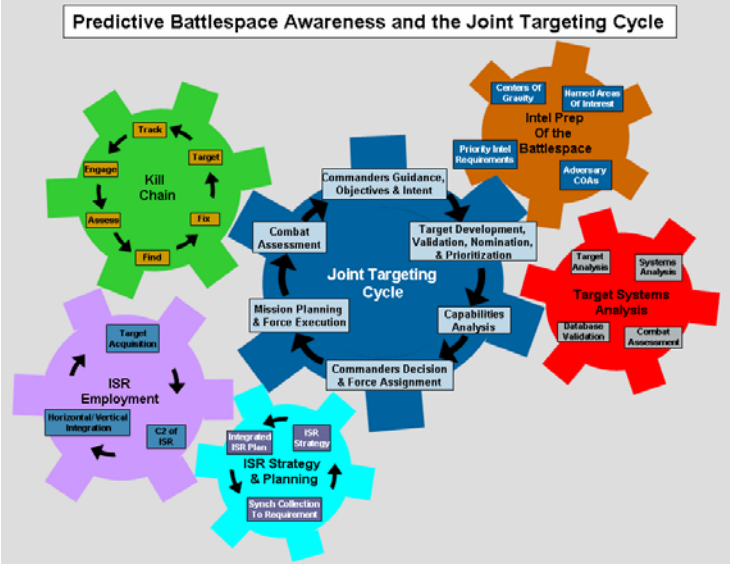


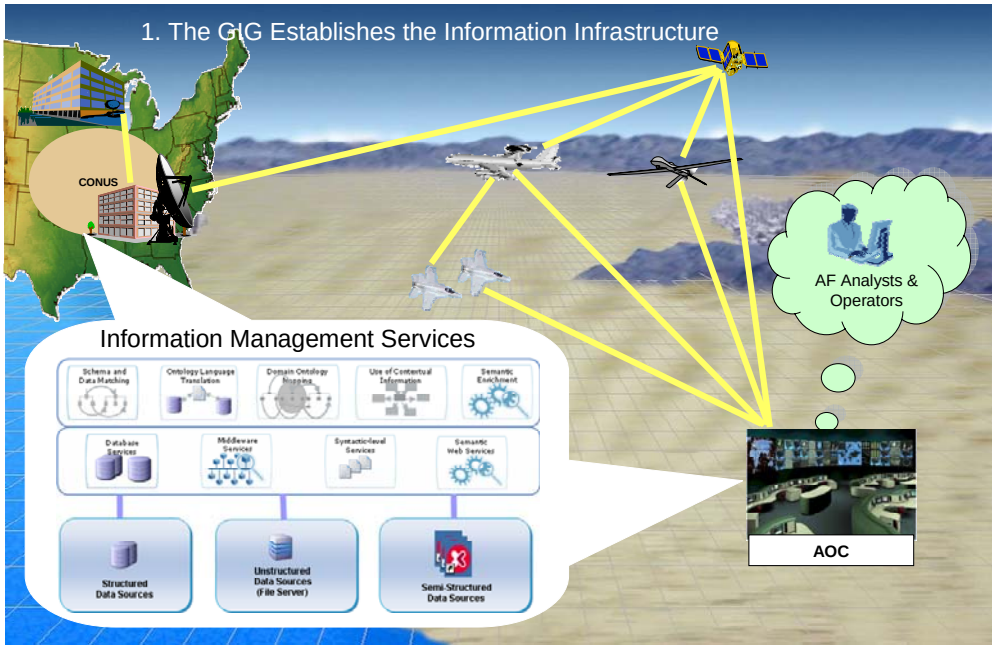
Figure 1: PBA consists of interrelated steps

As is shown in Figure 1, the steps of PBA are not completely independent. Information, or work products from one step can feed as input to another step (or multiple other steps). Semantic Interoperability technologies can certainly help with the automation of the process. In our associated 'MAISSI Final Presentation Brief' we describe the classes of technologies that are applicable for each step of the process. In this final report, however, we are focusing on the first step of the process, Intelligence Preparation of the Battlespace, and provide concrete examples that demonstrate the value of the SI technologies in a manner similar to the one use by the SAB-TR-05-03 Report on Domain Integration study.

The emergence of the Global Information Grid (GIG) yields unprecedented information access and sharing among Commanders, operators, Communities of Interest (COIs) and eventually the war-fighters. SI technologies enable these various constituents to effectively exploit the GIG Infrastructure by managing information efficiently to be of direct use to their operations. In particular, SI technologies facilitate information management along five major categories (a) Data Discovery/Context, (b) Mediation to multiple formats, (c) Alignment and

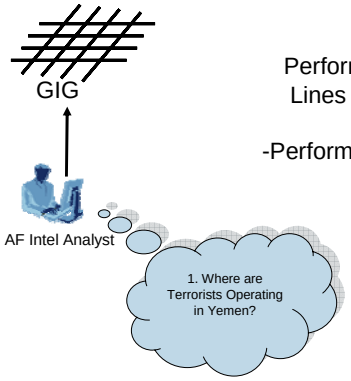
Matching across different semantic representations, (d) Reasoning, and (e) Data Transformations to the format of the downstream applications.

This use case demonstrates how these classes of SI technologies assist with the first step of the PBA process, the Intelligence Preparation of the Battlespace.

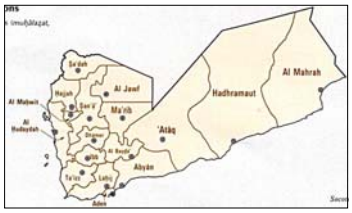


Step 1: The GIG establishes the Architecture which will be used by the JFACC's Air Operations Center (AOC). Operators and Analysts in the AOC are linked via tactical and satellite communications links to Air Force Command, ISR, and Strike assets as well as back to National and Air Force facilities in the Continental United States. These Gateways provide access to DoD and other Government Communities of Interest Data Bases and file shares while also providing access to open source material via the World Wide Web. Clients on the GIG can use a variety of software applications and information management services to access, retrieve, and synthesize data from a broad set of sources.

2. AF Intel Analyst begins IPB for a New Mission



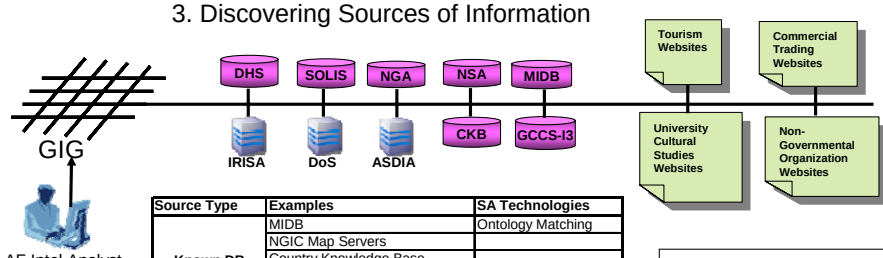
Receive JFACC Mission:
Perform ISR Operations to find Terrorist bases and Lines Of Communication in the Country of Yemen
First PBA Task:
-Perform Intelligence Preparation of the Battlespace for Terrorists in Yemen



Analyst inputs Lines of Inquiry into IPB application → metadata is added to the request → Use of Context decomposes request into Info searches

Step 2: Upon receipt of new JFACC Guidance and Directives in support of a new Mission, AOC Intelligence Analysts begin performing Predictive Battlespace Awareness Activities to drive the Joint Targeting Cycle and the Air Operations Campaign.

3. Discovering Sources of Information



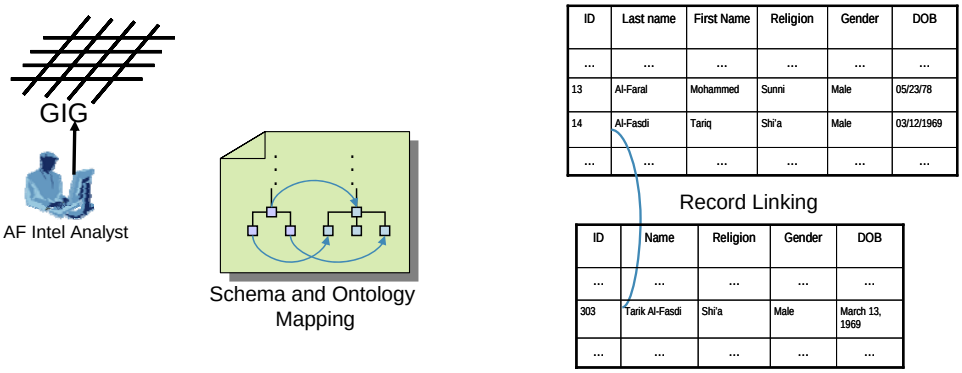
Source Type	Examples	SA Technologies
Known DB	MIDB	Ontology Matching
	NGIC Map Servers	
	Country Knowledge Base	
File Servers	IRISA	Schema Matching
	All Source Document Index	Data Matching
	NSA SIGINT	Context Discovery
	Department Of State	
Open Source	Yemeni Tourist Websites	Data Matching
	Univeristy Websites	Context Discovery
	Non-Governmental Organization Websites	

LEGEND
NGA – National Geospatial Intelligence Agency
CKB – Country Knowledge Base
NSA – National Security Agency
DHS – DIA Defense HUMINT Service
IRISA – Intelligence Report Index Summary File
MIDB – Modernized Integrated DB
SOLIS – SIGINT Online Information Service
DoS – Department of State
GCCS-I3 – Global Command and Control System Integrated Imagery and Intelligence
ASDIA – All Source Document Index

IS and SA technologies discover sources of information via the GIG Computer and Communications Infrastructure → Semantic Discover technologies identify relevant sources from established databases, file servers, and unstructured sources such as web pages

Step 3: The GIG provides access to known databases with defined ontologies, Community of Interest File Servers, and open sources such as Foreign Country and University Web pages which may be of interest. Semantic Alignment and Information Services Technologies discover and sources of interest to the query.

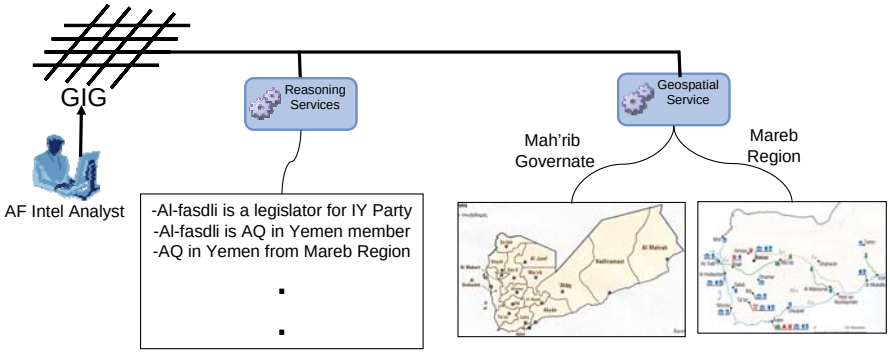
4. Correlate the information from relevant sources



Upon return of information from each source, alignment technologies correlate the source information ⟶ Schema and Ontology Mappings are established ⟶ data matching technologies link records from various sources

Step 4: Semantic Alignment and Information Services Technologies discover and match ontologies, Schemas and data to the data elements resident in the Intel Analysts query.

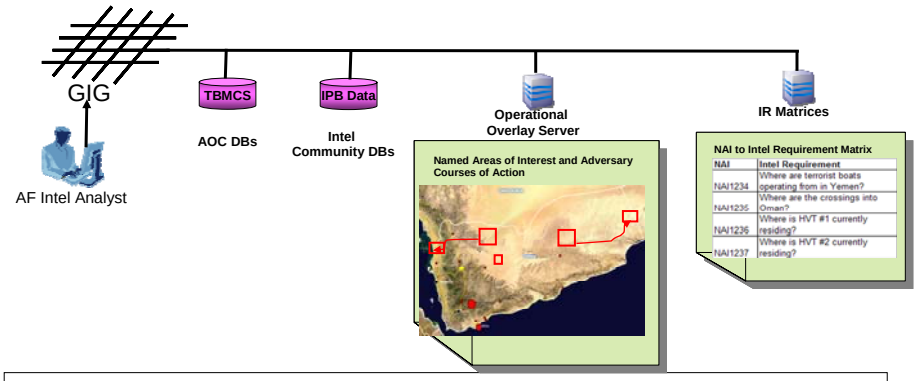
5. Transform the Information into a useful response to the query



Upon correlating the source data ⟶ Reasoning technologies transform the data into information relevant to the IPB task ⟶ discovery and use of geospatial services assist reasoning about the proximate distance between people and events in time and space

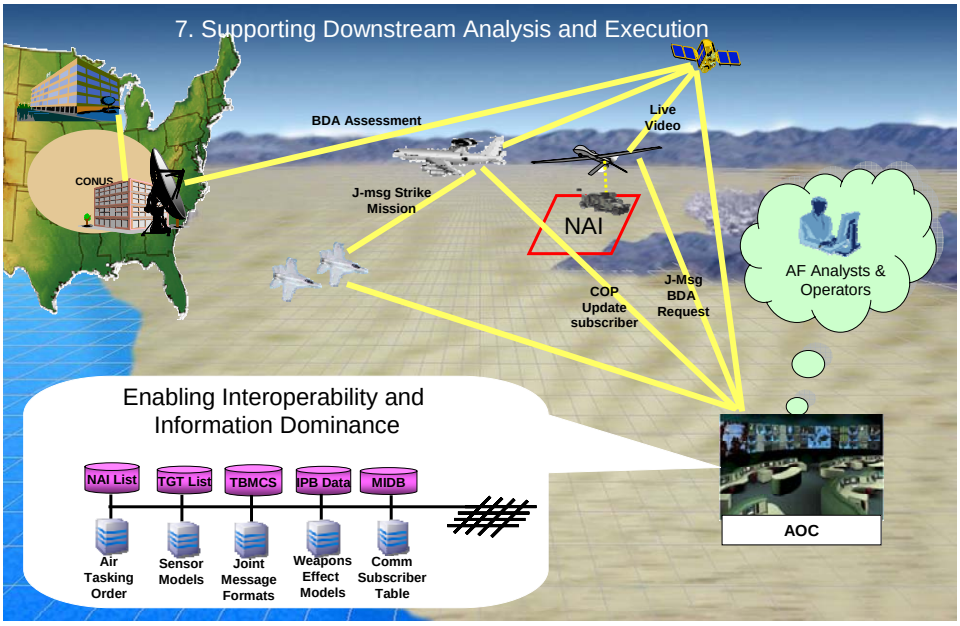
Step 5: Reasoners apply logic over the collected source data in tandem with their knowledge bases to semantically enrich the data. Information Services provide for the use of geospatial and geolocation services to verify the proximity to places and events to each other.

6. Publish Information into Formats Usable by Downstream analysts and operators



Outputs from IPB are then generated and published into databases and files which support down stream functions ⟶ Ontology and schema mapping from the original sources to the published databases can be maintained for information provenance

Step 6: The results of the C2 applications are finally stored in the databases and the ontology and schema mapping technologies facilitate the transformation to the format of the receiving data store.



In summary, as the figure above shows, Semantic Alignment and Information Services Technologies enable the discovery and transformation of information into formats and storage to support downstream application of the information by the right user on the GIG.

4 SEMANTIC ALIGNMENT TECHNOLOGY

This section discusses SI technologies addressing the semantic alignment challenge described in Section 2.2. Our trade study investigates technologies along the four category areas listed in Table 1 below.

Table 1: Semantic Alignment techniques are needed across a wide spectrum of information representations with varying expressive power.

Alignment Techniques	Capabilities
Schema and Ontology Matching	Discover mapping between database schemas. Build rules that convert data from one syntactic representation into another.
Data Matching/Record Linkage	Correlate instance level data. Deconflict data instances with similar characteristics
Use of Contextual Information	Extract meaning by looking at the context in which a given term is being used.
Semantic Enrichment	Enrich the data with derived entities and relationship in support of schema evolution and dynamic transformations.

4.1 SCHEMA/ONTOLOGY MATCHING

Mediation enables multiple services to exchange data based on a common vocabulary that guarantees consistent interpretation and behavior, while allowing for specialized knowledge and reasoning at the local level.

As defined in [16], mediators handle information exchange between a source and a receiver system in two steps: (a) queries from the receiver system are translated to equivalent queries for the source schema; and (b) the resulting source data are translated to the specification of the receiver system. This mechanism is shown in Figure 2, below.

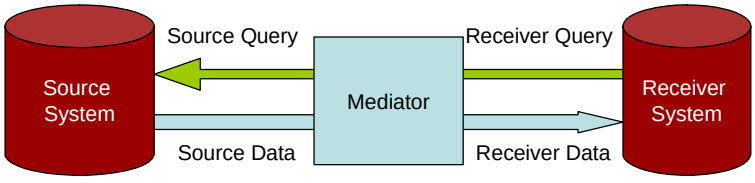


Figure 2. Semantic mediators translate between different data representations.

A preliminary step to query translation is that of *Schema Matching*. Elements from the source schema need to be mapped to entities on the target schema, and the pertinent relationships among the objects in the source model need to be translated to the representations of the target model. Despite the active research in this field, the scarcity of meta-data information and the non-standard use of model constraints often require human intervention. We explore the following automated techniques for Schema Matching: (a) *rule-based*, and (b) *machine-learning*. The rule-based solutions exploit schema information such as element names, data types, structure, foreign-key relationship and integrity constraints to determine the representation of the data

sources, and use handcrafted rules to map elements from one specification to another. Machine-learning techniques use statistical methods to build probabilistic models of the schemas, or analyze the values of the data elements to discover the semantics [13].

4.1.1 The Space of Schema/Ontology Matchers

There are a number of dimensions along which schema/ontology matchers can be categorized. These include:

- Architecture
  - Single algorithm
  - Hybrid: Fixed combination of algorithms
  - Composite: Flexibly combines different algorithms
- Algorithm types
  - Rules
  - Statistical learning
  - Probabilistic matching
- Automated vs. user intervention
- Similarity measures
  - Linguistic
  - Structural
  - Instances
- Mappings
  - Simple: 1 to 1 mapping of schema elements
  - Complex: map a schema element to a composition of multiple elements
- Ontologies vs. Schemas

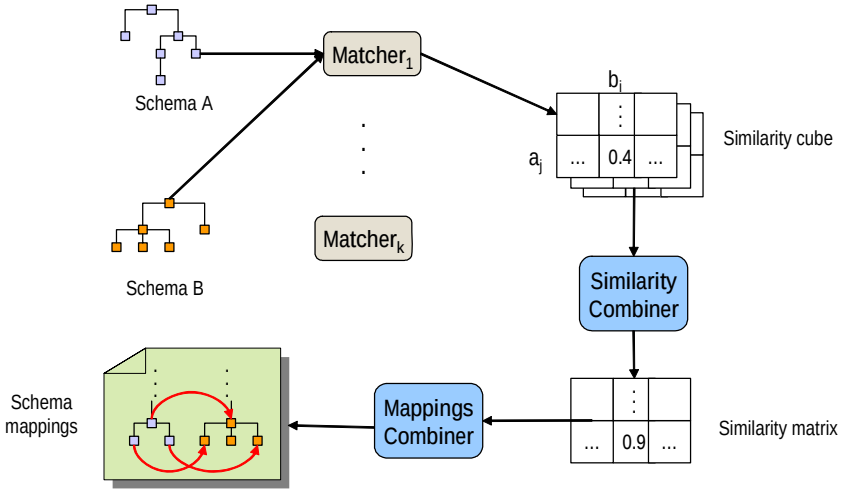


Figure 3. Composite matcher architecture.

Architecture. Schema matchers can be categorized as simple, hybrid, or composite. A simple schema matcher uses a single similarity measure to compute matches. A hybrid matcher (the most common type of matcher) combines multiple similarity measures, typically according to a fixed formula, into a single overall similarity measure. A composite matcher provides the

most flexibility, by combining the results of several independently operating matchers, each of which may be simple or hybrid. Different component matchers may be chosen depending upon the match task at hand and different combination techniques may be chosen. Figure 3 shows the architecture for a composite schema matcher. The processing steps for a composite matcher are:

- Each *matcher* outputs a similarity value for each pair of objects.
- The similarity values for each pair of objects for each matcher are put in the *similarity cube.*
- The *similarity combiner* combines the output of each matcher for each pair into a single similarity value for that pair.
- The *mappings combiner* produces a mapping from schema A to schema B that optimizes some function of the similarity values of mapped objects (e.g. the sum of similarity values)

Algorithm types. The algorithms used by schema/ontology matchers can be roughly divided into three types. *Rule-based* algorithms make use of "if, then" rules to determine matches. For example, one might have a rule that says that element X of schema A matches element Y of schema B if neither X nor Y has not been previously matched to any other element and the overall similarity of X to Y is greater than X's similarity to any other unmatched element. (This rule in effect implements a greedy matching algorithm.) Matching rules may be constructed manually or learned from examples.

In contrast to rule-based inference, *probabilistic matching* computes matches through some form of probabilistic inference. Bayesian networks, neural nets, or statistical classifiers may be used to do probabilistic matching. Typically, a probabilistic matcher is learned through training examples.

Statistical learning algorithms are used by some matchers to learn rules or networks for doing matching. An example is decision tree learning in which a set of manually constructed matches is given to the statistical learner, which produces a decision tree. Statistical learning is also used by some systems (i.e. iMAP) at a higher level to determine what weight to give to lower level matching algorithms in particular circumstances.

Automated vs. user intervention. Some matchers are completely automated and don't permit human intervention whereas others do permit human input into the matching process or even rely upon human intervention to a significant degree. Given the limitations of the state of the art in schema matching, the ability to integrate human advice into the matching process is crucial. Matchers that accept human input include COMA++ and iMAP (both discussed below).

Similarity measures. Nearly all schema/ontology matchers base matches on a measure of similarity between schema/ontology elements. Hybrid and composite matchers work by combining similarity values produced by single algorithms. Similarity may be measured along multiple dimensions. *Linguistic similarity* involves similarity in the terms used to denote elements in a schema or ontology. Two terms may be similar at a surface level by virtue of being composed of similar sequences of characters. Edit distance between two terms is a common measure of their similarity as character sequences. (Edit distance measures the number of alterations needed to turn one string into another.) *Phonetic similarity* looks at how the two terms are pronounced – i.e. at their similarity as sequences of phonemes. Finally, another type of linguistic similarity involves relations of synonymy. If two terms are synonyms (e.g. "fast" and "rapid") that is some evidence that they match (i.e. have the same meaning as used in the two

schemas). Being synonyms does not guarantee a match, however, since many words have multiple meanings. (E.g. "fast" might mean "go without food" rather than "rapid.")

Structural similarity measures look at the hierarchical structure of a schema or ontology to determine similarity of terms. When a schema or ontology is represented as a tree, where the edges in the tree represent either object containment or class/subclass relations, one take the representation of an element in a schema to be, not just the term in the schema for that element, but the entire path from the root of the tree to that term. A structural similarity measure computes the similarity of two terms in different schemas by comparing the paths to these two terms. Similarity of the two terms themselves plays a role in the path similarity but considering paths instead of single terms allows other terms in the two schemas to play a role in determining the similarity of the given terms.

Instance level similarity (or *extensional similarity*) is similarity of the instances of two schema elements. For example, one schema might have a term "ComputerManufacturer" with instances {Hewlett-Packard, Dell, Apple, IBM, Sony, ...} while another schema might have the term "Constructeur_d_Ordinateurs" with instances {Bull, LaCie, IBM, Dell, ...}. The two sets of instances might be similar in terms of having overlapping membership (IBM, Dell) or in terms of similarity of distinct instances. (E.g. both the Bull and IBM corporate websites mention things like servers and software.)

Simple vs. complex mappings. An important distinction between matchers is whether or not complex mappings are supported. A simple mapping is a 1-1 mapping between terms in two schemas. (E.g. "Plant" in one schema corresponds to "Facility" in another.) A complex mapping, however, involves a correspondence between a term in one schema and a combination of terms in another.

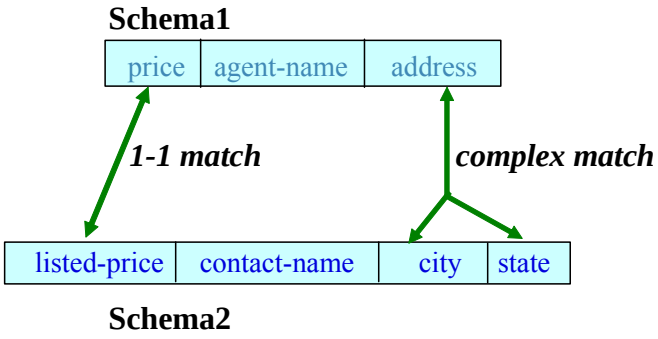


Figure 4. Simple and complex mappings.

Figure 4 shows examples of both types of mapping. "price" in one schema maps directly to "listed-price" in the other. However, there is no one term in Schema2 that corresponds exactly with the term "address" in Schema 1. Rather the term "address" in Schema1 maps to a function of two terms in Schema2 – the concatenation of "city" and "state."

Most matchers only try to find simple 1-1 matches between terms in two schemas. This greatly simplifies the matching task since the search space for 1-1 matches is finite. The search space for complex matches, however, is *infinite* since there are an infinite number of ways of combining schema elements. As will be discussed below, finding complex matches is a serious

challenge but cannot be ignored since there is often not a simple 1-1 mapping between each term in one schema and a term in another schema.

4.1.2 Schemas vs. Ontologies

The distinction between schemas and ontologies and the relevance of this distinction to semantic alignment merits discussion up front. A *schema* (such as a relational database schema or an XML schema) structures information and provides an intuitive semantics for data by describing attributes of data types and containment relations among data types (e.g. zip code is a component of address)

By *ontology* we mean *formal ontology* – i.e. an ontology described in a formal language such as first-order logic or OWL. Formal ontology languages have more built-in semantics than do schemas, making possible greater expressivity and enabling deeper reasoning with data expressed in an ontology.

Although there is some overlap in the research communities for schema and ontology matching, there are significant differences between the two communities. Although some matchers can be applied to both schemas and ontologies, many well known schema matchers fail to take advantage of the richer semantic information in ontologies. We have examined matchers initially tailored for both database schemas and for ontologies and discuss examples of each below. While the use of ontologies in principle allows richer information to be brought to bear in matching, special issues arise with regard to their construction, acceptance, and maintenance, which we will discuss in a separate section.

4.1.3 Performance Measures for Matchers

Recall and precision are common metrics for measuring matcher performance. Recall measures how well a matcher does at finding correct matches; precision measure how well a matcher avoids incorrect matches. Recall can precision can be defined in terms of three quantities:

#Right = the number of reported matches that are correct matches

#Wrong = the number of reported matches that are *not* correct matches

#Missing = the number of correct matches that are not reported

These statistics require access to ground truth.

Recall is #Right ÷ (#Right + #Missing) (i.e. the proportion of correct matches that are reported). Precision is #Right ÷ (#Right + #Wrong) ((i.e. the proportion of reported matches that are correct).

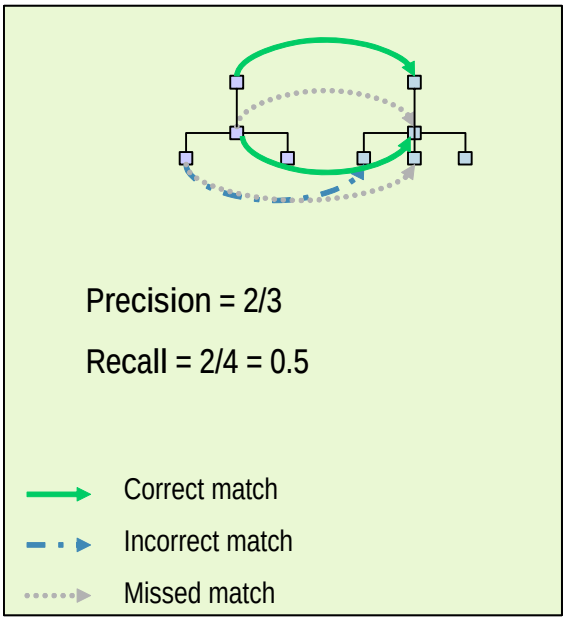


Figure 5. Precision and recall for schema matching.

Figure 5 shows an example of how precision and recall are computed.

Recall and precision are often combined into a single measure called F-measure. F-measure is defined by:

$$\text{F–measure}(\alpha) = \frac{\text{Precision} * \text{Recall}}{(1-\alpha) * \text{Precision} + \alpha * \text{Recall}}$$

α is an adjustable parameter that determines how much weight to give to recall or precision. If $\alpha = 1$, then F-measure becomes precision; if $\alpha = 0$, then F-measure equals recall.

Some researchers in schema matching have proposed a performance metric that reflects the saving of effort involved in correcting reported matches by adding misses and removing wrong matches ([20]). This metric is sometimes called "Overall" and is defined by:

$$\text{Overall} = \frac{\#\text{Right} - \#\text{Wrong}}{\#\text{Right} + \#\text{Missing}}$$

This measure achieves a maximum of 1 when #Wrong = #Missing = 0. The measure can be negative (there can be more wrongs than rights) and has no lower bound. It is somewhat difficult to interpret but one can see that increasing #Wrong or #Missing does decrease Overall when #Right > #Wrong. Anomalously, when #Right = #Wrong, #Missing has no affect on Overall. (Increasing #Missing should increase the effort involved in adding missed matches.) Even more strangely, when #Wrong > #Right, increasing #Missing *increases* Overall.

Despite these oddities, the Overall metric is on the right track: we do need a metric that measures the saving of effort in manual alignment. Other metrics with more regular behavior can be fashioned. For example, if we let t(S,T) be the time required to manually align ontologies S and T without any automated help and t(M, S, T) the time required with the automatically produced mapping M, then we could use

$$\frac{t(S, T) - t(M, S, T)}{t(S, T) + t(M, S, T)}$$

as a measure of the saving of effort. This ranges from -1 to 1, with 0 indicating no saving of effort, a negative value increased effort, and a positive value decreased effort. t(S, T) and t(M, S, T) can be estimated from characteristics of S, T, and M.

4.1.4 Schema and Ontology Matching Tools

This section surveys some representation schema and ontology matching systems. The aim of this section is not to provide a comprehensive evaluation of all the matchers out there but rather to use selected examples to illustrate the state of the art in matching technology and to probe some of the limitations of current matchers. The results of our evaluation of these matchers provide the data for the roadmap for future research that we will present.

Automatch. This matcher is of interest as an example of a matcher that uses probabilistic matching (**Error! Reference source not found.**). Automatch is an instance based matcher: it determines the similarity between two attributes by looking at the overlap in their instances.

Automatch assumes that a database schema is simply a set $\{A_1, ..., A_n\}$ of *attributes* and that each attribute has a set of values (instances) that belong to the attribute. An example is the domain of computer retail manufacturing and sales, where the attributes include *DesktopManufacturer*, *MonitorManufacturer*, *DesktopModel*, etc. The values of, for example, *DesktopManufacturer* would be members of the set {Apple, Compaq, Dell, ...}.

Attribute Dictionary	
Attributes	Values
A_1	$\{v_{11}, v_{12}, \ldots, v_{1k_1}\}$
A_2	$\{v_{21}, v_{22}, \ldots, v_{2k_2}\}$
...	...
A_n	$\{v_{n1}, v_{n2}, \ldots, v_{nk_n}\}$

Figure 6. Automatch's attribute dictionary.

Automatch requires an *attribute dictionary*, as shown in Figure 6, which is a list of possible values for each attribute. The attribute dictionary is compiled from previous schema matches by human experts. All known values of attributes matched by human experts to attribute A_i will be in the set of possible values of A_i.

Matches between two schemas are computed in Automatch by using the attribute dictionary as an intermediary. For each schema, the probability of an element in the schema matching an attribute in the attribute dictionary is computed and these probabilities are combined to determine a best match between elements of the two schemas.

To compute the probability that an attribute X in a given schema matches an attribute A in the attribute dictionary, Automatch assumes that for each value v, there is an unconditional probability $P(v \in X)$ that v will be observed to be a value of X and a conditional probability $P(v \in X \mid X \leftrightarrow A)$ that v will be observed as a value of X given that X matches A. Automatch further assumes that if $V = \{v_1, ..., v_k\}$ is a set of values, then $P(V \subseteq X \mid X \leftrightarrow A) = P(v_1 \in X \mid X \leftrightarrow A) \ldots P(v_k \in X \mid X \leftrightarrow A)$. (I.e., whether a value is observed to belong to X is probabilistically independent of what other values are observed for X given that X matches A.)

Given these assumptions, the probability that X matches A, given observed values for X can be computed as:

$$P(X \leftrightarrow A \mid V \subseteq X) = P(V \subseteq X \mid X \leftrightarrow A)P(X \leftrightarrow A)/ P(V \subseteq X).$$

This formula is a straightforward instance of Bayes theorem, which follows from the definition of conditional probability.

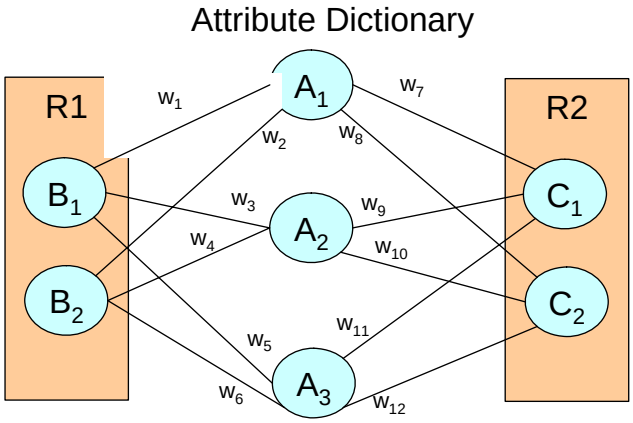


Figure 7. Matching weights between two schemas and the attribute dictionary.

As shown in Figure 7, given two schemas R1 and R2, a tripartite graph is formed in which each element in one schema is linked to each attribute in the attribute dictionary by a weighted link. The weight on the link is precisely the probability that the schema element and the attribute match given observed values for the schema element. Automatch uses this tripartite graph to find a "best" match between elements of R1 and elements of R2 by computing a path between each element of R1 and a unique element of R2 such that the sum of the weights on all paths is maximal. Automatch uses an ingenious method to find the matchings that maximize the sum of link weights: it turns the matching problem into a maximal flow problem with constraints and solves the maximal flow problem using a standard algorithm for such problems.

Discussion of Automatch. Automatch uses a sound probabilistic inference technique to assign weights to the links in the tripartite graph used for matching. The maximal flow algorithm is guaranteed to find a matching that maximizes the sum of the weights over links in the paths. However, the assumption that the goal is to maximize the sum of the weights over the links is equivalent to the assumption that the attributes in each schema are independent of one another, which is not in general true. For example, If in schema R2, attribute C2 is a subtype of attribute C1 (e.g. C1 = *ComputerManufacturer* and C2 = *DesktopManufacturer*), then which attributes of R1 match C2 is *not* independent of which attributes of R1 match C1. If B1 matches

C1, then only a subtype of B1 can match C2 – e.g. we would not want to match a supertype of B1 with C2.

One reason Automatch does not take into account dependencies among attributes is that it uses a very simplified representation of attributes: attributes are essentially atoms, with no internal structure. In particular, there are no hierarchical relations among attributes, as there is in schemas or ontologies that are more than lists of concepts. This is a serious limitation of Automatch, since it prevents it from making use of relational information about attributes that can be important in determining matches.

COMA++. In the first half of the Trade Study, we focused our attention on COMA++, a state of the art composite matcher ([22]). COMA++ has several advantages as a tool for studying schema matching technologies:

- A demo version is freely available for download
- It builds on a number of well-known previous hybrid and composite matchers
- It comes with an extensive library of matching algorithms
- It's matching algorithms cover all the main types of simple matchers
- It permits the flexible combination of individual matchers
- It performed well in the 2006 Ontology Matching Workshop, despite it's not being specifically designed to deal with ontologies (it was developed for relational database and XML schemas)

Matcher Type	Matcher	Schema Info	Aux. Info
Simple	*Affix*	Element names	_
	N-gram	Element names	_
	Soundex	Element names	_
	EditDistance	Element names	_
	Synonym	Element names	Extern. dictionaries
	Data Type	Data types	Type compatibility
	UserFeedback	-	User-specified (mis-) matches
Hybrid	*Name*	Element names	_
	NamePath	Names+Paths	_
	TypeName	Data types+Names	_
	Children	Child elements	_
	Leaves	Leaf elements	_
Reuse-oriented	*Schema*	-	Existing matches

Figure 8. Matching algorithms available in COMA++.

COMA++ has a large number of built-in algorithms for matching (see Figure 8). It provides several functions for combining similarity values obtained from different matchers:

- **Max**: return the maximal similarity value for any matcher

- **Weighted**: return the weighted sum of similarity values
- **Average**: Special case of Weighted in which weights for each matcher are equal
- **Min**: return the minimal similarity value for any matcher

We installed COMA++ and ran it on some several schema and ontology pairs. The purpose of the runs was to gain familiarity with schema matching techniques and to get an initial impression of the scope and limitations of those techniques. No systematic testing has been done yet. There is no doubt that the composite matching approach implemented by COMA++ is a significant advance over previous matching techniques. Nonetheless, we believe the approach taken by COMA++ has certain limitations and that further improvements in schema matching are possible.

First, it should be recognized that the use of similarity measures in determining matches is only a proxy for what we are seeking, namely, the probability that a term in one schema is semantically equivalent (or has some other semantic relation) to a term in the other schema. It is certainly true that certain kinds of similarity provide evidence of semantic equivalence or relatedness. However, the precise relationship between similarity along a certain dimension and probability of semantic equivalence is unclear.

Even if we assume that we could specify a probability distribution relating similarity along some dimension with probability of semantic equivalence, the combination rules used by COMA++ do not correspond to any coherent ways of combining probabilities. Assume, for example, that a similarity value for two terms can be interpreted as the probability that those two terms are semantically equivalent. (This is not necessarily a correct interpretation, but our critique does not depend upon the precise way in which similarity and probability are related.) If matcher M1 reports a similarity value for terms T1 and T2 of p, we may interpret that information as a conditional probability: $P(T1 \equiv T2 \mid M1)$ -- the probability that T1 and T2 are semantically equivalent given the output of M1. Suppose that a different matcher M2 yields a similarity value of q for T1 and T2. If we assume that the outputs of M1 and M2 are probabilistically independent of one another given the true state of equivalence or non-equivalence of T1 and T2, then we can calculate that the probability of T1 and T2 being semantically equivalent given the outputs of both M1 and M2 is given by:

$$P(T1 \equiv T2 \mid M1, M2) = \frac{pq}{pq + (1-p)(1-q)P(T1 \equiv T2)/(1-P(T1 \equiv T2))}$$

where $P(T1 \equiv T2)$ is the prior probability of semantic equivalence of T1 and T2.

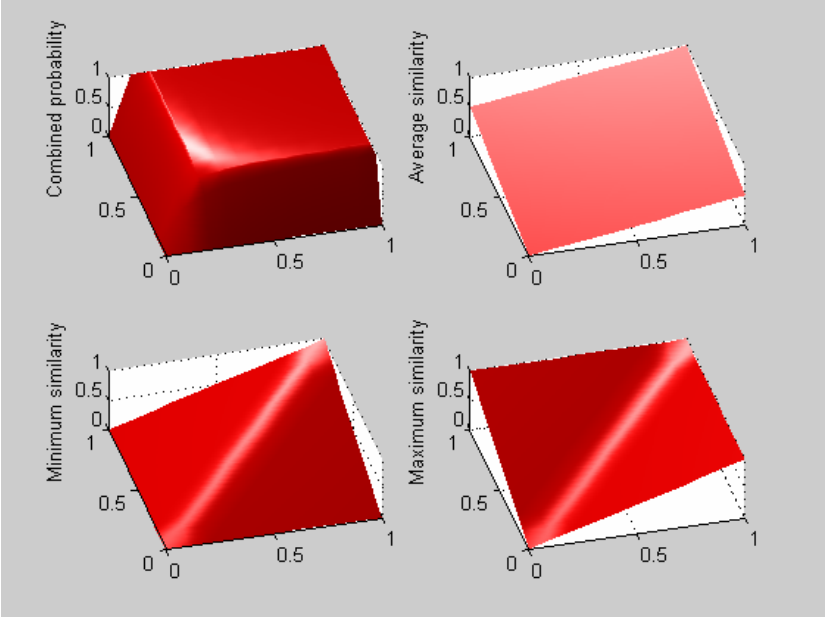


Figure 9. Combination functions for similarity values.

The top left graph in Figure 9 shows P(T1≡ T2 | M1, M2) (vertical axis) as a function of P(T1≡ T2 | M1) and P(T1≡ T2 | M2) (horizontal axes), where we have set P(T1≡ T2) to 0.01. This graph is to be contrasted with the other graphs, which show the average, minimum, and maximum combination functions for similarity values. It is clear from these graphs that none of the combination rules used by COMA++ corresponds at all closely to the probabilistic combination of values.

This is not to suggest that the particular probabilistic combination rule shown above is the correct one; however, these results suggest that there is no particular reason to suppose that the combination rules used by COMA++ correspond to any well-grounded way of combining probabilities of equivalence based on different matcher results.

The combination rules used by COMA++ also do not take into account variable correlations between outputs of different matchers – e.g. different matchers may be using same similarity measure or dependent similarity measures (character similarity and phonetic similarity), so their outputs will be correlated, or, they may be measuring similarity along completely independent dimensions, so that their outputs will not be correlated). If we are combining outputs from more than two matchers, the output of some pairs may be correlated while the output of other pairs may be uncorrelated. There is no way that a single combination rule of the sort used by COMA++ can adequately capture such variable correlations.

Evaluating COMA++. We did a simple experiment in which we used COMA++ to map each of two source ontologies into a target ontology. The results revealed some of the strengths and weaknesses of current schema matchers.

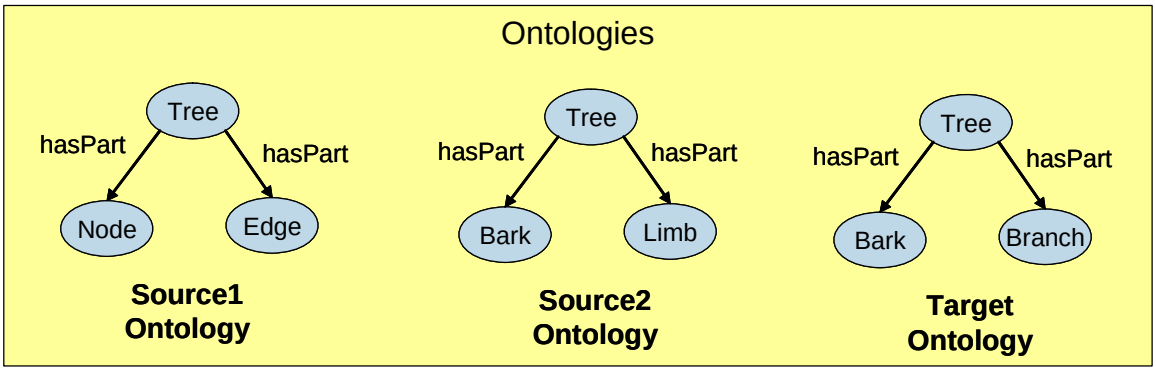


Figure 10. Test ontologies for COMA++.

Figure 10 shows the source ontologies and the target ontology into which they were mapped. Although these are trivial ontologies, the mappings discovered by COMA++ serve to illustrate issues that need to be considered in future schema matching research.

Source1/Target Mapping	Source2/Target Mapping
Source1:Edge ↔ Target:Branch: 0.868	Source2:Bark ↔ Target:Bark: 0.868
Source1:Tree.hasPart ↔ Target:Tree.hasPart: 0.82	Source2:Limb ↔ Target:Branch: 0.868
Source1:Tree ↔ Target:Tree: 0.82	Source2:Tree.hasPart ↔ Target:Tree.hasPart: 0.82
	Source2:Tree ↔ Target:Tree: 0.82

Figure 11. Mappings between source schemas and target schema.

We ran COMA++ on each pair of source and target ontology using the default setting for the AllContext matching strategy, which computes the similarity of each pair of paths in the two ontologies using name similarity, type of leaf nodes, and synonyms. The final similarity value for each pair of paths is the average of the similarity values from the constituent matchers.

Figure 11 shows the mappings discovered by COMA++. (We note that we manually added to the thesaurus that fact that "edge" and "branch" are synonyms (in graph theory) and "branch" and "limb" are synonyms.) The numbers after each mapping are the final similarity values for the two schema elements.

A number of issues arise from this simple example. First, although COMA++ computes a "best" mapping (meaning a mapping that maximizes the sum of similarity values), it gives no indication of the overall quality of the mapping. Although the mapping between Source1 and Target has one fewer correspondence than that between Source2 and Target, the similarity values for individual schema element pairs are comparable. Second, as mentioned above, similarity values are difficult to interpret and their correspondence to probability of match is obscure. Finally, it is apparent from this example that COMA++ does not make use of *context* information in determining the quality of individual or overall matches. By context information, we mean in this case information about the domain for which the ontology was created. In the case of Source1, a human looking at these ontologies could probably guess that Source1 was created for the domain of graph theory or perhaps computer science whereas the domains of Source2 and Target are probably the same. This information has relevance to the probability that terms in two of these ontologies match but there seems to be no way for COMA++ to make use of such

information. The issue of context arises in other areas of semantic alignment and we will devote a separate section below to discussing the use of context information.

LSD. LSD (Learning Source Description) is an example of a hybrid matcher that uses statistical learner to train component matchers.

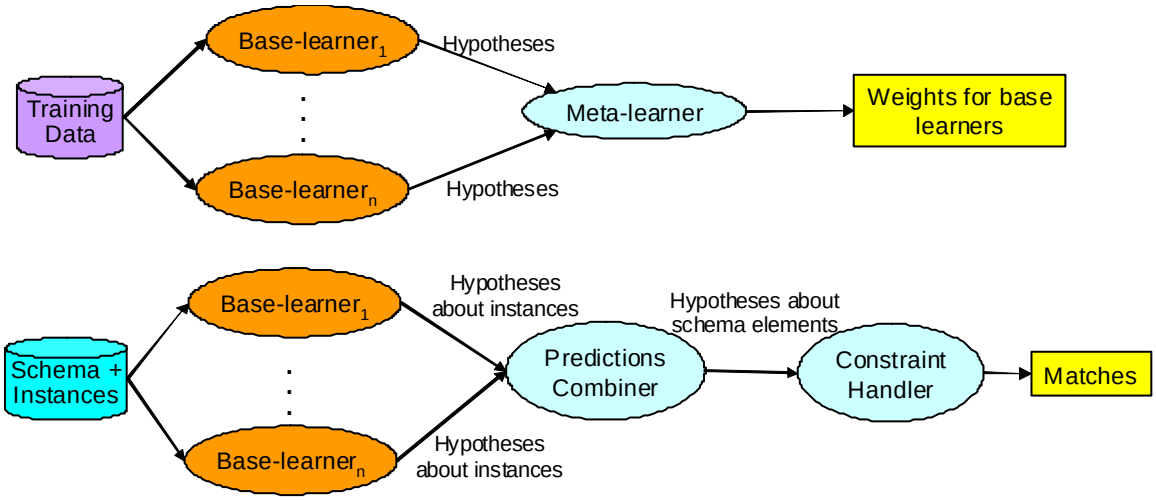


Figure 12. Architecture of LSD.

The base learners of LSD are trained on manually created matches. A meta-learner then assigns weights to the base learners based on their performance. The weights are relative to characteristics of the schemas to be matched. (E.g. in schemas with a rich hierarchical structure, more weight may be given to a matcher that exploits structural information.) The component matchers are essentially *classifiers*: they group instances of one schema into the categories of another. These classifications by different matchers are then combined to produce a mapping between the two schemas. LSD can make use of known constraints on schema elements. For example, if a relation in one schema is known to be functional (e.g. "motherOf"), it will not be matched to a relation in the other schema that is not functional.

In evaluations ([23]), LSD has achieved 71-92% accuracy. The evaluations demonstrate that the meta-learner significantly improves matching accuracy. LSD's base matchers are inadequate for certain types of schema elements, however. For example, the instances of the course code element in one schema tested were short alpha-numeric strings. The base matchers had difficulty classifying these instances, but in principle a format learner could have been used to do the classification.

iMAP. iMAP advances the state of the art in schema matching by tackling the challenging problem of finding complex matches. A complex match, recall, is one in which an element of one schema maps to a function of multiple elements in the other (e.g. "name" in one schema might map to the concatenation of "first_name", "last_name" in another).

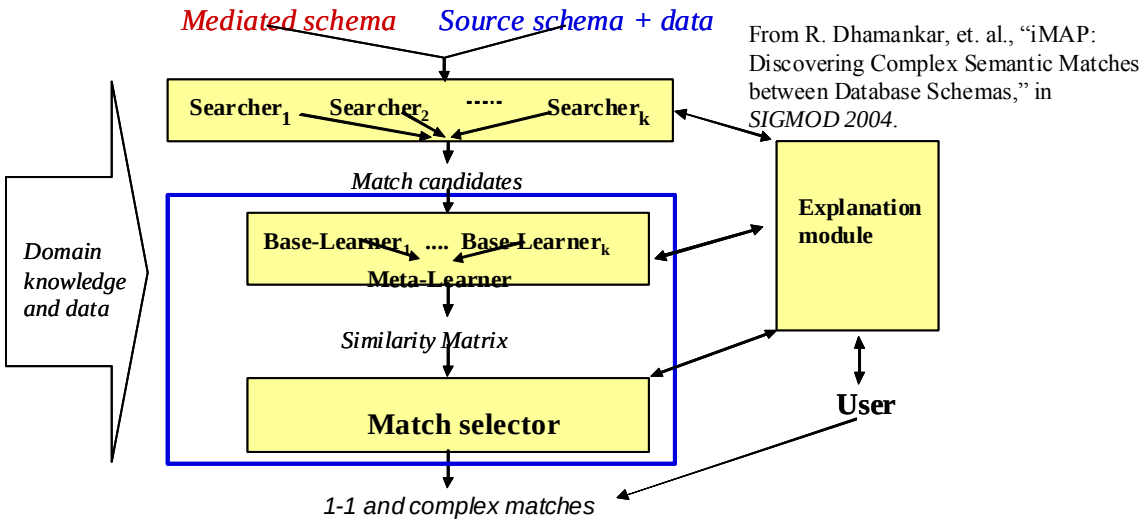


Figure 13. iMAP architecture.

Since the search space for complex matches is infinite, iMAP uses heuristics to search that space for plausible matches. These candidate matches are fed to a modified version of LSD, which outputs a set of 1-1 and complex matches.

Another novel feature of iMAP is its ability to give explanations for the matches found. Being able to explain results is important in gaining user trust in the system.

iMAP also allows user input in matching. The user may suggest particular 1-1 matches, eliminate candidate matches, or suggest the use of particular combination functions for complex matches. It turns out that considerable user intervention is required for iMAP to successfully generate complex matches.

4.1.5 Ontology Matching Evaluation Initiative

A 2002 review of schema matching algorithms ([20]) found it hard to make comparisons for several reasons

- Different test sets were used by different research groups
- Protocols were poorly described
- Dataset size varied considerably across groups
- Different metrics were used to evaluate performance

In recent years, these problems have been addressed by systematic evaluations of ontology matchers under the auspices of the Ontology Alignment Evaluation Initiative . The goals of this initiative, as described in its website, are:

- assessing the strength and weakness of alignment/matching systems
- comparing performance of techniques
- increasing communication among algorithm developers

- improving evaluation techniques

- **most of all, helping improve the work on ontology alignment/matching**

Evaluations have been performed yearly since 2003. The latest evaluation for which results were published in time to be analyzed for this trade study was the November 2005 evaluations ([21]). In 2005, ontologies from three ontology suites were tested:

-Benchmark test suite: bibliography ontologies

-Generated from a single ontology by systematic modifications

-Anatomy ontologies manually created by two medical research groups

-Web site directories

-Semi-automatically pruned versions of web site directories used by Google, Yahoo, and Looksmart

The matching systems evaluated in 2005 were:

- FOAM (University of Karlsruhe, Germany)
- OLA (University of Montreal/INRIA, Canada)
- CtxMatch 2 (IRST Trento, Italy)
- Falcon (Southeast University, Nanjin, China)
- Unnamed (UC. Dublin, Ireland)
- OMAP (CNR/Pisa, Italy)

The performance measures used in this evaluation were recall and precision. The results for the different test suites were as follows:

-Benchmark tests (Bibliography domain)
  - Best precision score: 0.91 – Falcon
  - Best recall score: 0.89 (Falcon)
  - Lowest precision: 0.08 – ctxMatch2
  - Lowest recall: 0.18 – CMS

-Web directories domain
  - Best recall: 0.32 – OLA best, Falcon close second
  - Worst recall: 0.094 – ctxMatch2
  - The ground truth data generated could not guarantee *completeness* of the ground truth mappings, so precision could not be calculated

-Anatomy domain
  - Results were not available at time of publication of results article

These results show a wide variance in performance among matchers and across test suites. For example, Falcon obtained very high recall and precision on the benchmark tests but had poor recall in the Web directories domain.

While the OAEI is a major step in the direction of systematic evaluation of matchers, a number of issues need to be addressed in order to properly interpret the evaluation results.

One issue is whether the test cases are representative of real world matching problems of concern to the Air Force. For each test suite, the ontologies are known to be from same domain. Moreover, the bibliography and anatomy ontologies are likely to use very standardized vocabulary. These features of the test suites are often not present in real world matching problems, where the domains of data sources may not be known, standard vocabulary may not exists, and matching might be done between schemas from different but overlapping domains.

A second issue is that the test protocols are not entirely clear. Organizers say in their summary paper that the "… graphs [of participants results] are not totally faithful to the algorithms because participants have cut their results (in order to get high overall precision and recall)" The meaning of this statement is not entirely clear. An e-mail inquiry to the organizers produced no response.

Finally, as mentioned previously, since some human intervention in matching is necessary for the foreseeable future, metrics that measure the savings in manual alignment effort may give a more useful picture of matcher performance than standard recall and precision metrics.

4.2 DATA MATCHING

Schema/ontology matching involves discovering correspondences between elements at the *type* level – i.e. between terms denoting classes or sorts of things. Data matching, or record linkage, involves finding correspondences between specific data instances – e.g. between a record for a terrorist suspect in one database and a record in another.

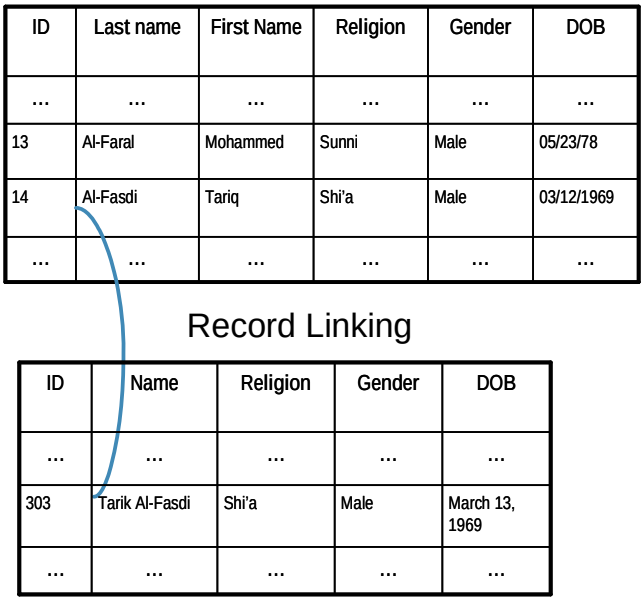


Figure 14. Data matching.

Figure 14 shows an example of data matching. Two databases with different schemas contain information about terrorist suspects. The entry in one database with Last name = "Al-Fasdi" and First Name = "Tariq" has been matched to an entry in the other database with Name

= “Tarak Al-Fasdi.” Information from these two records can therefore be merged into a common database.

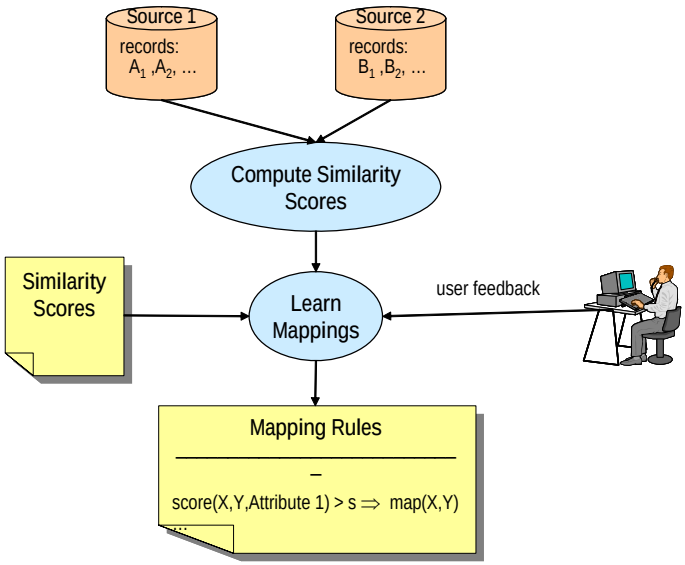


Figure 15. Active Atlas architecture.

The Active Atlas record linkage system [17] is a state of the art system for data merging. It employs user feedback on predicted matches to do decision tree learning of mapping rules. The mapping rules state when two data instances should be merged given similarity scores along certain dimensions.

Issues analogous to those in schema/ontology matching arise with regard to data matching. Although decision tree learning allows more complex rules to be learned than are used in a schema matcher such as COMA++, the use of similarity as a basis for matching may suffer from the same limitations as were seen in schema matching. The rules learned are deterministic; investigation of probabilistic inference in data matching may prove fruitful. Finally, as is common in schema matching, context information is rarely used.

Active Atlas has been extended to a system called “Apollo,” which makes use of secondary information to resolve entity references. For example, two records for a restaurant in different databases may have the same information except that the area code for one is different from the area code for the other. By consulting an external database containing information about area code changes, Apollo might be able to determine that one area code was changed to the other, verifying that the two records are indeed for the same restaurant. Apollo also makes use of a geographic database containing lat/long information for addresses.

The use of secondary information has been shown to produce significant increases in data matching precision and recall [17]. In one experiment, precision was 100 percent and recall was 76 percent with only 150 training examples when secondary information was used, but was lower when secondary information was not used, even though 250 training examples were used.

Other approaches to data merging include a variety of algorithms for name comparison (e.g. phonetic abstraction, Ngram indexing, Burkhart-Keller trees, partition filtering). Because these use only syntactic or phonetic features of names, they are often used as a first-pass filter for record linkage systems that make use of richer semantic information (e.g. attributes of entities).

Other systems for data merging include MARLIN, which uses support vector machines to learn similarity measures over strings; and PROM, which uses attribute similarity for entities, and information about the disjointness of attributes, to do entity resolution.

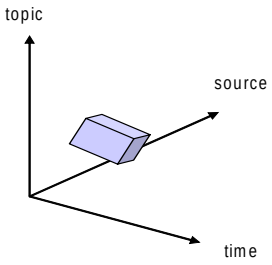
4.3 USE OF CONTEXTUAL INFORMATION

We have seen several areas in which context information is important to semantic alignment. In this section, we explore more fully what context is and how it is, or should be, used in semantic alignment. We start with an overview of some ways of understanding context and representing it. We then consider the use of context in discovering schema/ontology mappings. Finally, we look at the role context plays in query and data transformation.

4.3.1 What is Context?

Context has been understood in many different ways by different researchers. "context" has been used to mean, among other things:

- Metadata
- Structure of concepts within which a term in an ontology is embedded
- The spatio-temporal situation in which the ontology was created or used
- That portion of reality (or unreality) modeled by an ontology
- Purpose for which the ontology was designed
- Physical arrangement of labels in an interface



A useful general framework for representing context is to view a context as some portion of a multi-dimensional space whose axes as dimensions such as *source, time, purpose, topic, location*, etc. (Kashyap and Sheth, 1996). The specification of a region may be partial (e.g. a time interval) so that contexts are volumes rather than points in this space. While this is a useful picture, it should be noted that some of the dimension of context space, such as topic or source, have no natural ordering.

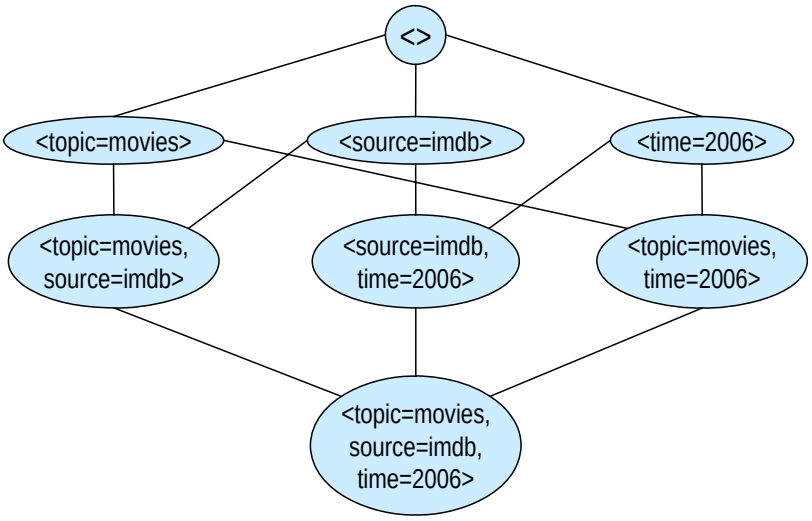


Figure 16. Context lattice for Internet Movie Database.

Given that contexts can be defined with greater or lesser degrees of specificity, they form a lattice, with the most general context ("everything") at the top of lattice and the bottom nodes being points in context space. In systems that reason with contexts, such as Cyc ([24]) and BAE/AIT's AIT Knowledge Server (AKS) ([9]), a context lattice (or tree in the case of AKS) is used to manage knowledge by partitioning the knowledge base into distinct but interconnected contexts. Information relevant to a given context is stored with that context and is inherited by all specializations of that context. Reasoning within a context is efficient since the inference engine need only consult that part of the knowledge base contained in the context or in more general contexts.

4.3.2 Contexts in Schema/Ontology Matching

Our survey of schema matching algorithms indicates that there is little or no use of context information in schema matching. Knowing that the contexts of two schemas are the same or different should influence our confidence in match results independently of particular similarity values. For example, if we know that two databases both deal with terrorist events, that should increase our confidence that, e.g., the term "perpetrator" and the term "organization" are synonyms in this context, over and above the evidence of semantic equivalence based on features internal to the two database schemas.

Context information can therefore be used to discover the semantics of data sources. If, however, the context of a data source is unknown, then there is a chicken and egg problem. If we knew the meaning of the schema, that would help us guess the context, but to figure out the meaning of the schema, we may need to know the context.

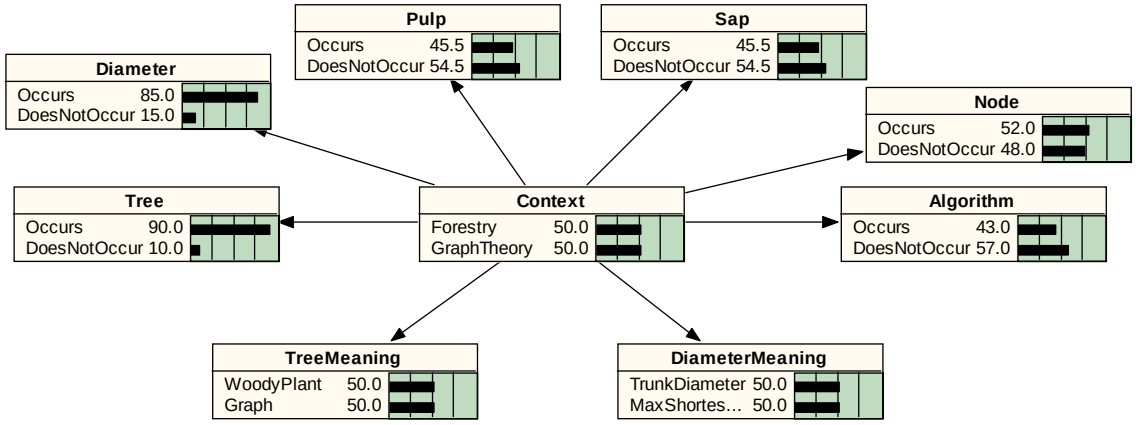


Figure 17. Bayesian network for context inference.

A probabilistic approach could be used to solve this problem. Here's an example of how this might work. The Bayesian network in the above figure shows how the meaning of certain terms depends on their context. But the probability of a term occurring in a schema also depends upon its context. If we find that certain terms occur in the schema, this gives us clues as to the context, which in turn provides evidence for the meaning of the terms. Here the fact the certain terms occur in the data source while others do not provides evidence about the context of the data source, which in turn influences our interpretation of the terms.

4.4 SPECIAL ISSUES REGARDING ONTOLOGIES

The vision of the semantic web is to leverage ontology languages such as OWL to enable machine processing of web content. This is a long range vision but important strides have been made already toward its realization. Increasing use of ontologies will impact how matching of representations is done and with what accuracy and efficiency. At the same time, the use of ontologies as a representation language carries with it some serious challenges that must be addressed before their benefits can be realized.

A recent workshop – the Ontology for the Intelligence Community Workshop -- addressed some of these challenges. A keynote address by Maureen Baginski, former Executive Director for Intelligence at the FBI, and former Director of the NSA Operations Center , made clear why there is a strong push in the intelligence community to make use of ontologies. Analysts are simply unable to keep all available information in their heads now. In the past, the focus of intelligence analysis was on collecting, not on search or sharing. Intelligence analysts analyzed what was dumped in their laps. The new paradigm, however, is one pull, not push. The analyst is actively asking, What do we need to know? What do we have now that might be relevant? Where can I find it? Who might have it? Ontologies allow analysts to ask the questions they want to ask, not be slaves to the mass of data collected. They do this by providing a way of navigating through vast amounts of data to locate and present in a meaningful way information needed by the analyst.

Unfortunately, ontologies are difficult beasts to tame. As Barry Smith, organizer of the workshop and director of the National Center for Ontological Research at the University of Buffalo, remarked, ontologies often reproduce the problems they are supposed to solve: people talking past one another, lack of application interoperability. Ontologies are often poorly constructed, by people who have little experience with ontologicial development. Such ontologies are an impediment to intercommunication, rather than a facilitator of it. Moreover, a Bill Anderson, chief scientist of Ontology Works, Inc., pointed out, one of the most serious challenges in the use of ontologies is ontology *maintenance*: life cycle maintenance of large ontologies is difficult because stewardship can often trump development costs. Terms may shift meaning, so that previously established correspondences between different ontologies are no longer valid. Ontologies may be extended in inconsistent or redundant ways, leading to further difficulties in stabilizing mappings.

Despite these difficulties, the consensus of the workshop was that ontologies, when carefully constructed and managed, provide benefits by providing key distinctions, constraints, and a durable integration schema.

4.5 SEMANTIC ENRICHMENT

Semantic enrichment refers to the augmentation of the underlying database schema or domain ontology with new concepts or relations, together with computational or inferential mechanisms for inferring instances of the new concepts and relations. Semantic enrichment can be used at the schema or ontology level to facilitate construction of a mapping between two schemas or ontologies, as is done in [14]; by embedding terms in two ontologies in a richer ontology, a similarity relation between terms in the two ontologies may be computed, which can then be used to establish a mapping between the ontologies.

Semantic enrichment also finds application at the data level as a means of augmenting the information in a database. Consider, for example, an intelligence database of information about infrastructure targets whose schema simply involves attributes of targets such as name, category, location and road segment data. An infrastructure analysis tool for analyzing connections between targets may make use of an ontology in which the key concept is that of *adjacency*. Semantic enrichment could infer adjacency between targets from the attribute information in the database by inferring that two targets are on the same road segment or connected by a short path of connected road segments with no intervening targets. By enriching the schema with the adjacency relation, the inferred instances of this relation can be added to the database.

Although semantic enrichment was not the focus of our investigations in MAISSI, we will be pursuing it further in the Tangram program. Tangram is a DARPA sponsored effort to integrate multiple group and link discovery tools in a grid environment. A common data representation is a crucial element of Tangram and semantic enrichment of the data will be needed in order for some of the components to handle the data.

4.6 ROADMAP FOR SEMANTIC ALIGNMENT

Our investigation of semantic alignment technologies has revealed a number of areas in which further progress can be made. In this section, we make a series of specific recommendations for how to further research in these areas.

Recommendation #1: Develop a Formal Theory of Matching

Current methods are ad hoc. Results can sound impressive for particular test sets but we lack assurance that the results will generalize to realistic matching problems. The notion of similarity needs to be put on a firm basis and its connection to probability of semantic equivalence demonstrated. A number of researchers hope for a clear probabilistic foundation for matching but no comprehensive framework is available,

Recommendation #2: Perform rigorous evaluations

Published performance results are often hard to compare due to use of different test cases and metrics. The Ontology Evaluation Initiative has made a good start at providing a uniform set of test cases and metrics for comparing matchers, but more realistic test cases are needed: Ontologies for different but overlapping domains should be used. A richer set of metrics is for evaluating matcher performance is needed. Precision and recall are important but they don't tell the whole story. Metrics that measure savings in manual alignment effort are needed, as well as utility based measures that look at how merged information is used in decision-making.

Recommendation #3: Exploit rich information for alignment

Whenever possible, contextual information should be used to help discover the semantics of the data. Key contextual parameters relevant to the semantics of terms need to be identified. Algorithms need to be developed that can infer the context of a source when the context is not already given and that can infer meaning from context. We need more research on how to exploit the richer semantics of ontologies, as contrasted with database schemas, to drive alignment.

Recommendation #4: Place emphasis on user-assisted semantic alignment, not on complete automation

Fully automated matching for complex, realistic ontologies is many years in the future. In the meantime, some human intervention is required to facilitate matching. The question is how to minimize human intervention. We need to discover bottlenecks in automated matching that can easily be overcome with human intervention. Since realistic matching requires the discovery of complex matches, users should be able to enter suggestions for combination functions and heuristics for searching for complex matches.

Recommendation #5: Develop an explanation capability for matchers.

Matching tools need to *explain* their choices to users before users will trust them.

5 INFORMATION SERVICES ARCHITECTURE

Cutting across the SI challenges of semantic alignment, data mediation and ontology transformation is the operational need to enable autonomous software applications to work together, and exchange information and services in a reliable and efficient manner. Multiple services for different abstraction layers are required to successfully integrate the data alignment capabilities into a comprehensive net-centric solution such as the GIG. In particular, we will explore information services addressing the four layers in Table 2 will be:

Table 2. Scalable SI requires information services spanning multiple layers of abstraction.

Service Layer	Required Capabilities
Database Services	Provide fundamental persistence, access, and update services for elementary data items. Support expressive queries and declarative rules for extracting, merging, and mapping low-level data to the Object Services layer.
Middleware Services	Provide communication protocols and APIs for object-level exchange. Includes distributed, cross-platform, and cross-language standards and support for multiple exchange patterns (e.g. client/server, publish/subscribe, peer-to-peer, etc.).
Syntactic-level Services	Provide platform- and language-neutral standards for syntactic data representation and exchange among coarse-grained systems. Provide a common operating environment supporting rapid addition and integration of new services through automatic registration and discovery.
Semantic Web Services	Provide semantically rich standards for describing properties and capabilities of new services. Support automatic composition of services.
Standards	Specifications that lead to integration through the use of common formats, languages and approaches

As with Semantic Alignment, technologies and standards at the lowest layers of the Information Services, such as Relational Databases and System Oriented Architectures, are relatively mature and widely used. The higher layers, however, still suffer from a plethora of emerging standards, incomplete specifications, and incompatible approaches for syntactic and semantic integration. This section describes some of the major technologies we plan to investigate in the area of Information Services, along these four major category areas.

5.1 DATABASE SERVICES

Databases are the most mature technologies in the area of Information Management. They abstract the mundane operations of data persistency, transaction management, storage fragmentation and data consistency, and provide a standard interface for consumer application to access and manage data. Their interfaces vary between three major modeling paradigms: *Relational*, *XML* and *Object-Oriented*. We will investigate leading tradeoffs inherent among these three interfaces in our trade study.

To address scalability issues and provide access to geographically dispersed sites, database vendors offer *clustered* and *federated* servers. Clustered Databases are usually employed in single site applications, where a large number of users collaborate over common high speed networks with low latency and high availability. Clusters use multiple processing units operating against common persistence devices (such as core memory and hard drives). Federated Databases, on the other hand, are more appropriate for Enterprise Systems with multi-site configurations, in which the data are distributed on completely independent servers. Federated Technology, for example, can be used in multi-strike planning missions between the AF CAOC centers and the Global Operations Center of STRATCOM.

To process data across multiple databases we will address in the trade study two major integration frameworks that are commonly employed: *Data Warehouse* and *Enterprise Information Integration*. In the former approach, the data from the source server are brought together into one single data warehouse, and in the latter approach, the data fragments are combined on demand into a virtual unified schema. The DW framework requires establishing the mappings and data scrubbing methods up front and providing the resources to load and manage the resultant large centralized database, while the EII framework incorporates dynamic data mediation and adaptive schema mapping techniques.

Database management is very important to the AF. In order to effectively integrate legacy Systems with newer applications, a proper combination of DW and EII techniques may need to be employed. As an example, consider the situation where Infrastructure data of adversary sites from the MIDB database are brought together with imagery data from Image Processing Libraries (IPL) and resource capabilities from the Defense Readiness Reporting System (DRRS) in order to plan a strike mission. Combining the information from all these databases into one warehouse will be time consuming and unnecessarily wasteful. A more appropriate technique will be to bring in only the relevant information from the Weather and Readiness systems depending on the location of the target and the types of capabilities required for the mission.

5.2 MIDDLEWARE SERVICES

Because of the maturity of database technology and their relative well understood interfaces, databases instances have proliferated in the DoD environment. This has resulted in replication of information across multiple sites and conflict of data among these instances. Consider, for example, the situation where an Air Operations Plan from TBMCS is imported in a collaboration system, such as GNCI or DSIDE [18], to be shared and manipulated by multiple collaborating COCOMS. The plan may be modified by the consumer systems to resolve temporal and resource constraints among its operational elements, but those changes may not necessarily be kept up to date with the source planning system. To address these challenges, collaborating applications employ the Middleware Services to exchange object level domain models among distributed applications.

Middleware Services offer meta-data management services to specify the interchange objects, and a programmatic interface to exchange the data. In MAISSI, we explored in particular the following commonly used exchange models: *client-server*, *publish/subscribe* and *peer to peer*. To utilize Middleware Services, applications either extend specific infrastructure classes that supply the needed processing logic, or invoke messaging classes to send and receive data and get notification when new information is available. In certain instances, the Middleware

Services supports persistence with location transparency through a vendor neutral, object oriented database abstraction layer.

Middleware is infrastructure software that is used to enable communication between applications (inter-process communication). Different portions of a single software application or component can easily communicate because they operate in the same portion of a computer's (virtual) memory. Typically this is accomplished by allowing different execution threads shared access to the same block of memory. Operating systems support this sharing of information across multiple threads with constructs such as mutexes and semaphores. Operating systems also provide support for sharing information across the memory boundaries of processes by using sockets, memory mapped files or pipes. Building distributed systems in terms of the low-level constructs provided by the operating systems is difficult and error-prone. Middleware is used to mitigate this problem by providing convenient abstraction layers over the facilities typically provided by the operating system. It forms the foundation for building service-oriented architectures and for enterprise integration.

Over the past twenty years different types of middleware have been developed to meet different aspects of enterprise integration. There is no middleware solution that can globally address all types of system integration challenges. Rather, system integrators must carefully weigh their system integration requirements against the costs and benefits of the different types of middleware solutions available.

In terms of how two applications exchange information, middleware can be characterized as *connection-oriented* or *message-oriented*. Connection-oriented middleware allows two applications to directly communicate by sharing a connection which is an abstraction of some low-level operating system construct such as a socket. Examples of such middleware are Common Object Request Broker (CORBA) and Java Remote Method Invocation (RMI). Message-oriented middleware typically introduce at least one level of indirection in the communication between processes by implementing it in terms of *messages* that are sent to the recipient through an intermediary. There are many commercial implementations of such middleware, many of them based on the Java Messaging Service (JMS) standard.

Services implemented in terms of the middleware can be:

1. *Public or Private*. Depending on whether they are available outside the system.
2. *Stateful or Stateless*. Depending on whether state is maintained across multiple invocations from a client.
3. *Coarse or fine grained*. Depending on the semantics of the information exposed.

Services are typically organized in a hierarchy (set of *tiers*), with top level services that are typically coarse grained and stateful. These top level services are typically implemented in terms of lower level services that are more data-oriented and fine grained. A service client is *external* if the network path across the middleware between the client and the invoked service is not totally controlled by a particular organization. Middleware issues that are affected when a client is external include firewall configuration and communication protocols (RMI/IIOP versus HTTP/HTTPS).

An important trade-off in selecting middleware is based on the observation that in general high interoperability (maximal loose coupling) can result in sub-optimal runtime performance. If high runtime performance is critical to the usability of the system, it may be necessary to select a middleware solution and to make architecture choices that may result in tighter coupling between

components. It may be necessary to adopt multiple types of middleware to address different aspects of the QoS requirements. For example a service may expose a coarse interface to external web clients and could be implemented internally using a high performance architecture based on CORBA.

Considerations that are important when selecting middleware include the following:

1. Amount of coupling between components. Message-oriented middleware generally induce looser coupling between components.
2. Data Throughput and other QoS guarantees. Connection-oriented middleware are better suited when large data rates between components are required.
3. Whether the application is distributed across a LAN, WAN or over the internet. Service Oriented Architectures implemented as Web Services are better suited when distribution over the internet is desired.
4. Middleware services. These include naming, trading or discovery, event management, transaction management and encryption. Different middleware choices have varying levels of standardization and maturity for such services.
5. Security

5.2.1 JMS Java Messaging Service

This is a standard developed by Sun Microsystems for message-oriented middleware for the Java platform. A Messaging Server (or a federated set of servers) is required to act as an intermediary for message routing. Applications can publish messages to *Queues* or *Topics* that are usually administered resources of the Messaging Servers. In systems that consist entirely of Java components a message can be a serialized Java object. JMS can be used for heterogeneous systems as well. For example, components written in the C++ programming language can be part of a JMS system by using a C++ to JMS bridge component that is commercially available. For such systems messages are typically text based, usually XML, complying with an agreed upon XSD. JMS is an excellent approach to building easy to configure, loosely coupled systems.

There are many implementations of JMS, including the reference implementation by Sun, as well as several open source implementations. JMS implementations can include persistence and guaranteed message delivery, high levels of reliability with fail-over, clustering and transaction support. JMS architectures are more suited for internal architectures. For external clients, JMS can be configured to utilize HTTP tunneling in order to cross firewalls.

Standard and Tool Developers: Standard development for JMS was lead by Sun Microsystems. Commercial implementations of the standard include: FioranoMQ, SonicMQ, Tibco EMS, WebSphereMQ, and Sun Java System Message Queue. Open source implementations include AciveMQ and JORAM.

Function: JMS is a standard for message-oriented interprocess communication.
Features: A JMS implementation includes a JMS server that acts as an intermediary between publisher and subscriber clients. A JMS implementation may have provisions for federating the JMS servers. A server can be persistent if it is configured to persist messages to a database, and a subscription can be durable if the subscriber is guaranteed to receive all information that appears in a topic of interest.
Strengths and Weaknesses: JMS is a mature standard with many high-quality commercial and open-source implementations. It forms the foundation for building more proprietary constructs such as Enterprise Service Bus (ESB) implementations. Although Java platform specific commercial and open-source products exist that allow C++ applications to become JMS clients. The learning curve for creating JMS clients is small.
Published performance results: Many, frequently contradictory performance comparisons can be found on the internet. As with other commercial technologies (e.g. RDBMs) it is easy in a comparison test to tune parameters in order to give one vendor an advantage over another. We looked at a number of such reports (see for example [19]) to derive an aggregate sense of the level of performance one can expect from a JMS-based system. Typical performance metrics include scalability of the server as the number of topics and clients (both publishers and subscribers) increases, and scalability of the server as the number of clients increases while keeping the number of topics fixed. These types of measurements can be obtained while allowing for persistent or non-persistent servers and durable or non-durable subscribers. The performance results can vary across different implementations by an order of magnitude. Furthermore, as we mentioned earlier, results from running a particular JMS can be highly sensitive to implementation-specific tunable parameters.

A rough sample of the expected performance of a JMS system based on an evaluation of four commercial products is as follows:

Assume two networked Windows PCs (3GHz, 2GB RAM) both running the Java HotSpot Client VM. The JMS server is running in one machine and all the clients (both publishers and subscribers) on the other. In the context of 10 topics, 10 publishers and 10 subscribers with a 1024 byte message size with non-persistent publishers, non-durable subscribers one can expect a message throughput of anything between 2,500 and 25,000 messages per second. In the presence of persistence and durability the throughput rate decreases to between 500 and 5,000 messages per second. The commercial systems tested showed good scalability as the load increased from 10 publishers and 10 subscribers to 50 publishers and 50 subscribers, by roughly maintaining their previous maximum throughput rate in steady state operation.

5.2.2 ICE Internet Communications Engine

The Internet Communications Engine (ICE) is a proprietary alternative to CORBA or the Microsoft analogs COM/DCOM/COM+. ICE is available both under GPL and under commercial license. ICE is an implementation of a proprietary specification that is a streamlined version of the CORBA specification. It is reputed to run faster and require less bandwidth than an equivalent CORBA implementation. The Slice language is the ICE analog of CORBA's IDL (see CORBA discussion below), and supports a simplified set of data types, without for example less frequently used but high overhead concepts such as type *Any*. ICE, like CORBA, has bindings to multiple popular languages, but most importantly it has a highly efficient binding to C++, in contrast to the complex and error-prone binding of CORBA to C++. ICE relies on a simple, more flexible proprietary communication protocol, instead of using IIOP. (See www.zeroc.com)

It may be a serious disadvantage that ICE is supported by a small private company.

Standard and Tool Developers: ICE is a proprietary standard and is currently implemented only by on vendor.

Function: ICE is a standard similar to CORBA for connection-oriented middleware.

Features: ICE has bindings to the following languages: Java, C++, C#, Visual Basic, Python, Ruby, and PHP.

Strengths and Weaknesses: ICE is easier to use, has a smaller footprint and a smaller learning curve than CORBA. ZeroC also claims that it is easier than CORBA to configure and deploy in wide are networks. It claims better runtime performance than CORBA. However, it appears that its user base is fairly small and it is championed by a small company.

Published performance results: The results mentioned have been conducted and published by ZeroC, the company that is both the author and implementer of the ICE standard. These are comparative tests between ICE and TAO, an open-source CORBA implementation, and were conducted both on Linux and on Windows XP.

Invocation latency is the time it takes for a client to complete a call to the server. Various types of invocations were measured (two way, two way asynchronous message, and one way). The results indicate that ICE is from around 100 to 200 percent faster.

Throughput experiments measure the ability of the middleware to transfer large amounts of data. In the case of transferring raw bytes, TAO has an advantage of 70% under Windows and 20% under Linux. In the case of sequences of structures ICE claims an advantage between 30 and 80%.

ZeroC has also published benchmark data comparing the TAO Event Service with the ICE Event Service (IceStorm), using various configurations. According to these ZeroC benchmarks IceStorm is typically 2 to 3 times faster than TAO.

Maturity: It is not clear how mature the ICE specification and the ICE standard are. ICE is currently in release 3.2. Boeing and SAIC have based their middleware solution for the Future Combat Systems (FCS) on ICE. Commercial customers include Skype. Government users include NAVSEA. The Naval Undersea Warfare Center uses ICE in its implementation of a Distributed Network Forces test environment.

Automation support

Compatibility and Applicability to the GIG specification: ICE would probably be a poor choice because it requires both the publisher and the subscriber to use ICE for communication. Furthermore it uses a proprietary protocol for data transport.

5.2.3 MQ Series

MQSeries is an IBM family of software products that together can be used as a middleware infrastructure for distributed application development.

MQSeries consists of three products:

- MQSeries Messaging, which provides the communication mechanism between applications on different platforms
- MQSeries Integrator, which centralizes and applies business operations rules
- MQSeries Workflow, which enables the capture, visualization, and automation of business processes

The point of business integration is to connect different computer systems, diverse geographical locations, and dissimilar IT infrastructures so that a seamless operation can be run. IBM's MQSeries supplies communications between applications, or between users and a set of applications on dissimilar systems. It has grown in popularity as applications are made available over the Internet because of its support of over 35 platforms and its ability to integrate disparate automation systems.

An additional helpful feature is that its messaging scheme requires the application that receives the message to confirm receipt. If no confirmation materializes, the message is re-sent by the MQSeries. IBM asserts that MQSeries can connect any two commercial systems that are in current business use.

Standard and Tool Developers: This is an IBM proprietary product. Websphere MQ implements the JMS standard.

Function: Message-oriented middleware with workflow capabilities.

Features

Strengths and Weaknesses: Provides support for C components.

Published performance results: See section on JMS performance results

Maturity: This is a mature commercial product.

Automation support: N/A

Compatibility and Applicability to the GIG specification MQ Series, along with other messaging implementations, is highly compatible with the GIG specification.

5.2.4 ESB

Enterprise Service Bus is an abstraction layer over an implementation of an enterprise-wide messaging system. It enables the construction message-oriented distributed applications using abstractions that hide the details of the underlying messaging infrastructure. Frequently, an ESB is used as a foundation for deploying Service Oriented Architectures and many vendors provide an ESB as part of their SOA offering.

ESBs are operating system and programming language agnostic, and allow communication between C++, Java and .NET applications. XML is used as a standard way to format messages. Frequently ESBs include support for Web Service orchestration as well as security implementations to authorize and authenticate. ESBs allow integration of legacy systems flexibly and efficiently in a standards-based manner. However, such integration does require that all systems be message-oriented, and messages can induce unintended coupling between the components. ESBs also introduce additional processing overhead over basic messaging systems.

Standard and Tool Developers

ESB is a description of a set of a capabilities rather than a standard. These capabilities can be implemented in different ways. ESB is often viewed as a proprietary product, but can also be seen as an architectural style that can be implemented in different ways.

Function

The following capabilities are generally accepted as part of an ESB:

- Invocation support (synchronous and asynchronous)
- Routing
- Mediation
- Process orchestration capability (for example BPEL)
- Complex event processing
- Quality of Service support (security, transactions, reliable delivery)
- System management (administration tools, logging, auditing, monitoring)

Complex event processing is the capability to process events that are frequently part of a larger sequence of events that might be required in order to complete a higher level event such as a transaction. This capability, in addition to event filtering, includes pattern matching on event streams. Process orchestration is the ability to execute different workflows of processes on the server side in order to satisfy either an external or an internal service request.

Features

Since ESB is a catch-all term, features vary by implementation.

Strengths and Weaknesses

There is no well-defined specification for an ESB, but there are multiple commercial products that implement ESBs. Most of these implementations are layered on top of mature technologies such as message-oriented middleware.

Published performance results

For performance estimates relating to the messaging portion of an ESB see section 1.1.1.5.

Maturity

Many of the commercial ESB products appear to be mature.

Automation support

N/A

Compatibility and Applicability to the GIG specification

Most of the technologies that define an ESB should be also relevant to the GIG.

5.3 SYNTACTIC-LEVEL SERVICES

Extending the interface mechanisms of the Middleware Services, Syntactic-Level Services enable autonomous software applications to register their services on the network, discover other applications with which to interface, authenticate and authorize users to access data on a per-

need basis, and exchange information via well-defined and agreed protocols. We compare the following three mature technologies for these services: *CORBA*, *J2EE* and *Web Services*.

5.3.1 CORBA

CORBA is a specification for connection-oriented middleware with many commercial and open source implementations and bindings to multiple programming languages. CORBA is an Object Management Group (OMG) standard with mature implementations of many associated middleware services including Naming Service, Event Service, Distributed Transactions, Trading Service (yellow pages for discovery of services), authentication and security. Furthermore, for systems with real (or near-real) time requirements, CORBA includes specifications for Quality of Service (QoS) that can be used to specify real-time requirements, and there are open source implementations of these Real Time extensions of CORBA (ACE/TAO). CORBA induces tighter coupling between interoperable components, because components must share a definition of a set of data structures that can be exchanged at runtime. These definitions are expressed in terms of CORBA's Interface Definition Language (IDL). CORBA is also problematic when processes are distributed across the internet. Although it is possible to configure CORBA components so that they can interact in the presence of firewalls, it is not straightforward to do so, and in practice CORBA has been limited to building systems that have defined high performance requirements and can be completely deployed in a LAN.

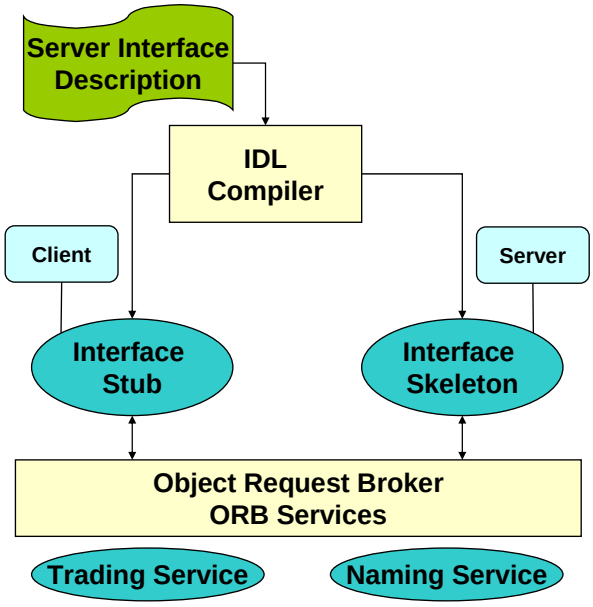


Figure 18. The OMG Object Management Architecture (OMA) reference model specifies a complete middleware infrastructure for distributed object management.

CORBA includes an architecture for distributed component computing known as the Object Management Architecture, shown in Figure 18. The scope of the OMA is comprehensive and somewhat complicated, which has also been an impediment to CORBA's adoption for certain software development efforts.

The OMA provides an architectural separation of concerns. *CORBA Services* provide standard life-cycle management services for objects such as creation, deletion, access control, events, persistence, transactions, queries, and time synchronization. *Vertical CORBA Facilities*

represent interfaces providing computing solutions for business problems within a specific vertical market (e.g., healthcare, manufacturing, finance). The OMG website provides lists of published and recently adopted Vertical CORBA Facility specifications. *Horizontal CORBA Facilities* represent those components providing support across an enterprise and across businesses. Finally, the *Application Objects* part of the architecture represents those application objects performing specific tasks for users. The OMA has influenced many DoD standardization initiatives and provides useful architectural guidance for the BIKE architecture.

U.S. military systems being developed today are required to use commercial and open standards, such as CORBA, to the greatest extent possible. The Joint Technical Architecture (JTA) specifies CORBA as the standard for developing and providing distributed computing services. Several military applications of CORBA are well-known because of their active participation in the OMG, such as the Navy's Open Computing Environment (OCE), the Joint Tactical Radio System (JTRS), the Defense Information Infrastructure Common Operating Environment (DII COE), and Air Force Theater Battle Management Core Systems (TBMCS) programs.

CORBA's infrastructure is useful for creating mechanisms for load balancing, resource control, or fault tolerance on the server side with great flexibility. However, the origins (Unix RPC), flexibility, and scope of the CORBA standard combine to create an extreme learning curve for developers. Key stakeholders designed CORBA by committee, which led to a wide representation of concerns in the standard, but sometimes too much specificity for unencumbered development. Because of these and other factors, such as higher network communication overhead, CORBA is often rejected as the middleware technology of choice, or it is "simplified" by non-standard implementations. Of all the computing platforms, CORBA is still the most widely used standard, but often as a hidden, enabling technology for system integration.

Standard and Tool Developers

CORBA is an OMG standard. There are more than two dozen commercial implementations and more than a dozen open source implementations. The commercial implementations include IONA, SUN and Visibroker. The open source ones include JacORB, TAO, and ORBit.

Function

CORBA is connection-oriented middleware for interprocess communication.

Features

CORBA supports vendor and platform neutral interprocess communication. It allows designers to create an object model for the information being exchanged as well as a description of the services being provided using the Interface Definition Language (IDL). A compiler for this language generates "stub" and "skeleton" code for the CORBA servers and clients that satisfy these interfaces. Cross-platform, cross-vendor interoperability is an important feature of CORBA. Components written in Java can interoperate with components written in C++ or any other language for which CORBA has a binding.

Strengths and Weaknesses

CORBA provides better QoS than Web Services. It has a steep learning curve, especially its C++ binding. It induces tighter coupling between components. It is mature and has

implementations of real-time extensions. However, it is more appropriate for deployment within a LAN or WAN.

Published performance results

CORBA includes a QoS specification (OMG Real-Time CORBA 1.0 specification). The open source TAO CORBA implementation includes an implementation of these real time extensions.

Maturity

CORBA is a mature, although complex, standard. There are also many mature implementations of the CORBA standard both commercial and open source.

Automation support

Compatibility and Applicability to the GIG specification

CORBA has limited or no applicability as an external service. Such services are typically exposed as Web Services. However, due to better QoS CORBA can be used in the second and third tier to address performance issues.

5.3.2 Web Services and J2EE

J2EE and Web Services, on the other hand, are refined though communities of interest, which is different from the OMG consortium. J2EE and Web services architecture involves many layered and interrelated technologies operating over the Internet, usually using HTTP, the transport protocol for web pages. Although J2EE and Web Services offer similar capabilities, they differ in the way applications can make use of their services. J2EE is based on the Java programming language and requires that all collaborating applications adhere to the Java associated technologies (such as Java Bean, JMS, WSDP, etc). The Web Services architecture, on the other hand, is technology agnostic and is based on standards such as XML, SOAP, WSDL, and UDDI to draw existing infrastructure together and provide a basis set of functions for interoperation, such as messaging, directories of business capabilities, and descriptions of technical services. This approach, however, has resulted in a proliferation of standards addressing specific distributed computing issues. As a guidepost to web service interoperability, the Web Services Interoperability Organization (WS-I) recently announced the release of the final specification of Basic Profile 1.0 a set of recommendations on how to use web service specifications to maximize interoperability.

Web services are self-contained business functions that operate over the Internet, usually using HTTP, the transport for web pages. Based on standards such as XML, SOAP, WSDL, and UDDI (described below), web services draw existing infrastructure together to provide a basis set of functions for interoperation, such as messaging, directories of business capabilities, and descriptions of technical services. Other functions are in active development.

Web services represent a vision of the World Wide Web as a bastion for electronic commerce. The canonical example for the future of web services is a complex, seamless commercial transaction such as airline reservations. A web service-based electronic commerce application is decomposed into independent, layered capabilities, conforming to W3C standards or recommendations. Figure 19 shows a generic web service architecture and the relationship among the communication, messaging, application interface, and process definition layers.

Though web services represent a vision of interoperability through standards, the process of standardization occurs though communities of interest, which is different from the OMG consortium. Many companies have been developing solutions in parallel with standardization efforts. This has resulted in a proliferation of standards addressing specific distributed computing issues. As a guidepost to web service interoperability, the Web Services Interoperability Organization (WS-I) recently announced the release of the final specification of Basic Profile 1.0 a set of recommendations on how to use web service specifications to maximize interoperability. The Basic Profile is an extremely detailed set of recommendations for web service practitioners who use the full gamut of standards. Below is a description of the most important standards which comprise web services.

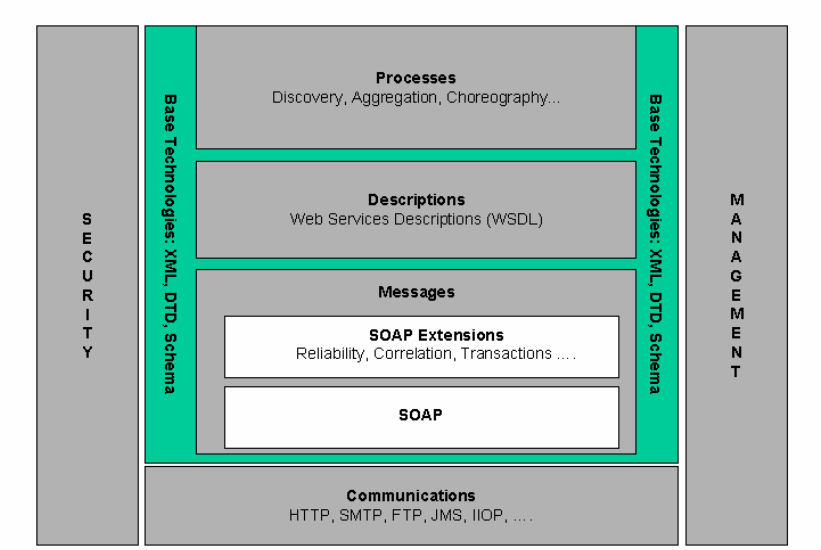


Figure 19 – The W3C Web Service Architecture consists of several layers of technology to enable interoperability across heterogeneous systems.

Standards and Frameworks

The following standards (current and under development) are relevant to the development of Web Services (in alphabetical order):

- BPEL4WS: Business Process Execution Language for Web Services defines a notation for specifying executable business interactions as well as describing the messages exchanged between two parties. Multiple additional standards either already exist or are in development in the area of WS Choreography. WS-CDL is the Web Services Choreography Description Language
- HTTPR: Reliable delivery of HTTPO packets between server and client that will enable reliable messaging between web services
- SAML: Security Assertion Markup Language is an XML framework for exchanging authentication and authorization information.
- SOAP: Simple Object Access Protocol enables XML-based messages that can be used to exchange structured and typed information between web services.
- UDDI: Universal Description, Discovery, and Integration is a specification for a SOAP-based web service that is used for locating web services based on WSDL-formatted

descriptions of the protocols for accessing these services. WS-Discovery is an emerging standard for a multi-cast service discovery protocol.

- WS-Addressing: This is a specification that enables message transmission in a transport-neutral manner through network nodes with endpoint managers, firewalls and gateways.
- WS-Coordination, WS-Atomic Transaction: Protocols for coordinating actions of distributed applications
- WSDL: Web service Description Language provides an XML-based protocol for describing Web services
- WS-Federation: WS Federation Language is a specification that defines protocols for different security realms to communicate by brokering trust, identities, attributes and authentication. Part of the WS Security Model
- WSIL: WS Inspection Language is a specification for an XML-formatted approach to inspect a site for available services
- WS-Security: Enhancements to SOAP messaging to ensure message integrity, confidentiality and authentication

The list of standards above is far from complete.

Web Services frameworks include the following:

- Apache Axis
- JSON-RPC Java
- Java Web Services Development Pack
- Web Services Invocation Framework
- Windows Communication Foundation
- XFire: a next generation Java-based SOAP framework that facilitates the implementation of high-performance Service-Oriented Architectures, through a simple, standards-based API. (http://xfire.codehaus.org)
- XML Interface for Network Services
- gSoap: A set of generator tools for coding SOAP/XML Web Services in C and C++. It provides a transparent SOAP API, and uses compiler technology to map XML schemas to C/C++ definitions. SOAP/XML interoperability is achieved with a simple API that hides the details of WSDL and SOAP.

Function

Web Services allow communications between applications across the internet. In particular, Web Services can be used to implement systems in accordance with principles of Service-oriented Architecture (SOA). Message-oriented services are services where the emphasis is on the message being exchanged rather than on the operation being performed.

RESTful Web Services are designed by analogy to protocols like HTTP by constraining their exposed interfaces to a set of standard operations with suggestive semantics such as GET, PUT, and DELETE. These services focus on client interactions with stateful resources rather than messages or operations.

Features

Applications are loosely coupled, since they are only bound by the format of the messages being exchanged.

Strengths and Weaknesses

Loose coupling and ease of use across the internet are strengths. Uncertain Quality of Service (QoS) in terms connection latency, and throughput for data intensive applications.

Published performance results

None found. There are many reports of poor or unpredictable runtime performance for data intensive applications or when QoS is an important aspect of the interaction.

Maturity

Standards are still being developed and evolving especially in the areas of security, messaging, transactions and orchestration.

Automation support

Standards such as BPEL4WS support automation.

Compatibility and Applicability to the GIG specification

Web services are the preferred approach to service development for the GIG.

5.3.3 Boeing SoSCOE

Of particular interest to the AF, is an emerging architecture from Boeing, called *System of Systems Common Operational Environment (SoSCOE)*. The SoSCOE has an architectural goal to present a full set of integrated C2 capabilities through the use of primary utility services called Task Integration Network (*TIN*). TIN-able application differ from the purely Service Oriented utilities in the way they are scheduled and tasked. The TIN framework provides a Hierarchical Task Network (HTN) logic to decompose and order the implementation of services based on C2 objectives and data interchange needs. Figure 20, below, shows the decomposition of the TIN-able services to support Mission Execution. Each one of the blocks in the diagram represents a service that is broken down to its sub-ordinate components. SoSCOE provides the framework to define TIN-able services and register the decomposition rules. The rules themselves are specified by the various domain planners (for example, maneuver, logistics, air support, etc.) and are developed by the appropriate service components.

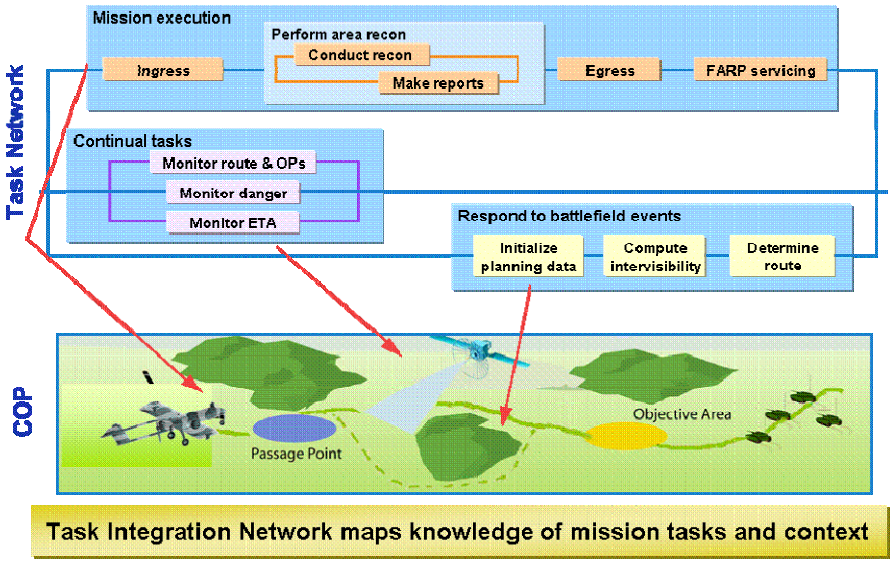


Figure 20: Mission Modeled In Task Integration Network

Like CORBA and Web Services, the FCS C2 architecture is service-based. A *service* is an independently compiled, stateless software unit that knows how to perform some foundation task(s). A service uses credentials (defined by a *role*) for data access and is intended to interact with FCS system database via the SoSCOE. The primary mechanism for organizing the services around roles is the *Task Integration Network* (TIN). A TIN is a logical arrangement of services (or utilities) organized in such a way as to achieve some useful user-oriented objective. It provides the thread(s) through all of the necessary services. (Note that the word *service* within the context of the TIN can also be used to include the word *utilities.)* TINs use a Hierarchical Task Network (HTN) application framework to decompose and order the implementation of services. This framework provides an extensible execution flow with robust error recovery and constrained execution support.

The FCS architecture is decomposed into mission applications, groupings of related services, including the Warfighter Machine Interface (WMI), Situation Understanding, Planning & Preparation, Battle Command & Mission Execution, and Sustainment. The FCS architecture is still in the conceptual design, and subcontracts have been awarded to General Dynamics and others to refine requirements and build mission application. Boeing is drawing on its subcontractor's expertise in distributed computing platforms, such General Dynamics' Openwings framework, to shape the design of the SoSCOE. BIKE research can draw from and contribute to this process. The BIKE project will help extend and refine the requirements of for the FCS information model, Battle Command semantics, service orchestration, and workflow policy definitions for the distributed Army and JIM perspectives.

5.4 SEMANTIC WEB SERVICES

Finally, at the top level of the technology stack are the Semantic Web services which extend the capabilities of Semantic Web ontologies with a core set of markup constructs for describing the properties and capabilities of their web services in unambiguous, computer-interpretable form. In our study we reviewed the following set of technologies: OWL, OWL-S and COIs,

Reasoners, Web 2.0 technologies, and finally the new term Web 3.0 that encompasses all of the above.

5.4.1 Ontology Web Language (OWL)

OWL is a semantic web language for describing classes, individuals, and their relations. OWL incorporates merges many of the representational constructs of description logics and frame systems. It uses an XML syntax, building on top of RDF, an XML-based language for describing web content. There are three dialects of OWL of increasing expressivity. *OWL Lite* provides a classification hierarchy (class/subclass relations), binary relations between objects, and simple constraints on relations (e.g. 0/1 cardinality constraints).

OWL DL ("DL" for "Description Logic") is the most expressive *decidable* dialect of OWL. (A logic is decidable if there is a mechanical procedure for determining whether or not an inference in the logic is correct.) OWL DL extends OWL Lite by adding functions for constructing new classes out of existing classes and a richer set of constraints on relations.

OWL Full has the maximum expressiveness. It extends OWL DL by allowing classes and properties to be individuals. That is, a class, such as Dog (the class of all dogs) can be treated as an individual by, e.g., being an instance of some higher order class such as Species (all of whose instances are classes).

5.4.2 OWL-S and COIs

The value of semantic interoperability cannot be fully realized unless flexible and adaptable architectures are in place for the composition of information services over the GIG. OWL-S [10] supports this capability by providing a framework to register service definitions and allow composite systems to evolve through object-oriented techniques. OWL-S describes services in three aspects: *profile* (set of functional capabilities provided), *model* (decomposition of the service into finer-grained steps), and *grounding* (list of interfaces). McIllraith [11] and Wu [12] describe techniques for using service models specified in OWL-S to automatically compose services.

In addition to these challenges, the Defense Information Systems Agency is mandating that combat systems be integrated in a net-centric environment in terms of dynamically composable services operating in the GIG. In this context, COIs evolve that are characterized by similar vocabularies and ontologies. In order to accomplish the goal of composability of services it is necessary to address the issue of semantic interoperability and semantic alignment extending across different COIs in areas where there is semantic overlap. The resulting interoperability is the foundation for network-centric warfare, improved situational awareness, and enhanced collaborative interaction among multiple joint and coalition platforms, sensors, units and individuals.

5.4.3 Reasoners

A big challenge in systems integration is to adapt software under changing technologies and evolving requirements. This becomes particularly important for enterprise-level applications that serve many constituents and interface with disparate systems. This could be accomplished using traditional repository technologies, i.e. relational databases, Java, C++, etc. For example, one

could write a custom application that retrieves the values of the objects from the database, determines their location and other characteristics, and based on the rules of engagement assesses their status as enemies of not. However, any application we build with procedural programming techniques will be specific to the current problem and will not be easily extensible. Similar work will have to be done when the situation changes, i.e. new rules of engagement. Semantic technologies make software more fit to survive in today's ecosystems of interacting components. This section describes newly emerging services for the semantic web that allow one to apply generic reasoners to modular rule bases that are separate from application software and are therefore modifiable and extensible without requiring changes to the underlying software.

We looked at several reasoners for OWL. Racer is a commercial reasoning system for OWL. In a comparison of Racer with a commercial database information retrieval system ([25]), the authors found that Racer is comparable in speed to the database system when the queries are simple data retrieval queries but for more complex queries, Racer is much slower. However, the greater speed of the database system comes at the expense of completeness of inference: the database system cannot find all valid inferences. The authors point out that an issue affecting all OWL reasoners, not just Racer, is the need to have all data in main memory in order to reason about it. This is a limitation that will have to be overcome if OWL reasoners are to scale to realistic problems.

Another study ([26]) compared Racer with two other OWL reasoners: Pellet, and FaCT. The authors found that Racer and FaCT are more efficient on classification tasks while

Pellet is faster than Racer in answering conjunctive queries over instance data (FaCT does not provide reasoning support for instances).

A number of lessons emerge from our examination of OWL reasoners. The reasoners tend to be tailored for academic research audiences and generally have poor documentation and a limited user interface. There is too often a focus on "logic-centric" concerns such as decidability at the expense of usability. Most OWL reasoners use tableau proof procedures that facilitate proofs of soundness and decidability but may not be as efficient as other proof procedures. We recommend more interaction between the OWL reasoner community and the logic programming community since the latter has decades of experience with compiling efficient programs (especially with regard to the Prolog programming language). Better knowledge management techniques are needed to overcome the limitation that all data must be in main memory to be reasoned with.

We also examined ways in which the expressivity of OWL can be extended to support richer representations and reasoning. SWRL is a language that extends OWL with the ability to expresses Horn clauses rules. (A Horn clause rule is an "if, then" rule with variables whose "if" clause is a conjunction of atomic statements and whose "then" clause is a single atomic statement.) Rules are of the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold.

The vocabulary for the rules is derived from an associated OWL ontology and the rules allow for the expression of constraints not expressible in OWL alone. Atoms in these rules can be of the form C(x), P(x,y), sameAs(x,y) or differentFrom(x,y), where C is an OWL description, P is an OWL property, and x,y are either variables, OWL individuals or OWL data values.

To facilitate with the understanding of SWRL and also demonstrate its use to the Air Force, we constructed a simple example in SWRL of rules that allow reasoning about whether an aircraft is friendly or enemy. The example illustrated the potential for semantic web reasoners to provide a flexible, adaptable framework for managing information needs.

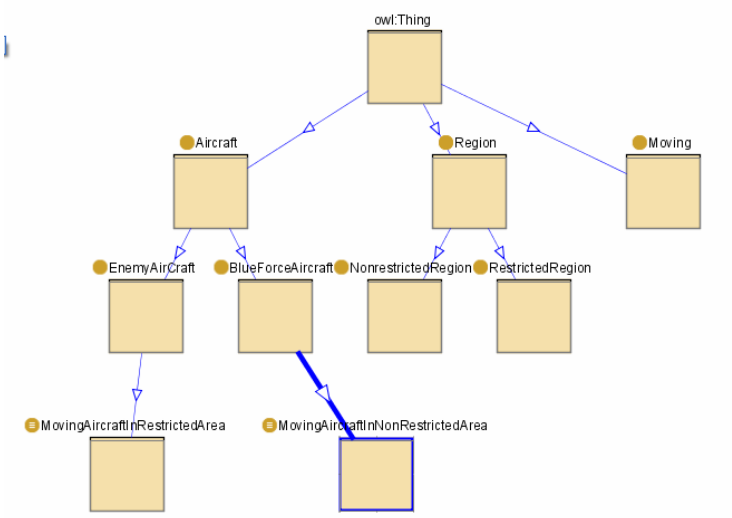
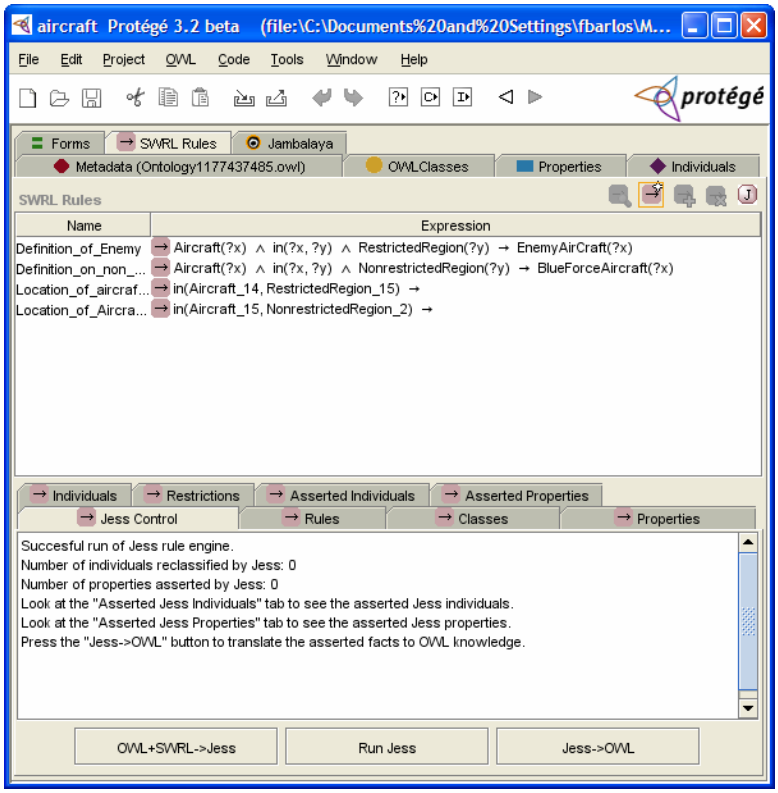


Figure 21: An ontology of aircrafts and regions

Consider, an ontology of aircrafts and regions, as shown in Figure 21. Aircrafts can be classified as enemy-aircrafts and blueForce-aircrafts. The ontology contains properties, 'in' and 'moving', that capture the information that an aircraft is located in a particular region and whether it is moving or not. The ontology shown was developed with the Protégé tool.



Now suppose that we extend this ontology with the following rules and facts:

- Definition of enemy: Any aircraft that is located in a restricted area is classified as an enemy aircraft
- Definition of BlueForce: Any aircraft that is located in a *non* restricted area is classified as a BlueForce aircraft
- Fact_1: Aircraft_14 is in restricted region 15
- Fact_2: Aircraft_15 is in non-restricted region 2

These rules are encoded in SWRL and are shown in the adjacent figure.

The benefits of Semantic Technologies is that, given the facts above and the rules in the ontology, a reasoner can infer that Aircraft_14 is an enemy aircraft without any custom coding or special purpose tools. As the rules of engagement change, new SWRL specifications can capture these extensions. For example, it will be very easy to add a rule that an enemy aircraft needs to be a moving object and rerun the reasoner to reclassify the aircrafts. More importantly, as we integrate with other services this automated system can provide a more complete operational picture and synchronize events with other commands and echelons. For example, if the objective is to reduce the risk of fratricide, we can add a rule not to prosecute targets that are very close to blue force units.

Although reasoners seem to provide great benefits, there come with severe limitations. Their use requires different sets of skills than traditional software engineering. Coding in logic is a completely different paradigm than procedural languages and testing and debugging is more difficult. Moreover, the tools are not as mature yet as traditional engineering applications.

5.4.4 Web 2.0

Web 2.0, a phrase coined by **O'Reilly Media** in **2004**,refers to a perceived second-**generation** of Web based communities and **hosted services** — such as **social networking sites**, **wikis** and **folksonomies** — that facilitate collaboration and sharing between users.

Web 2.0 technologies provide the following list of capabilities:

- Provides an adaptive semantic glue to form communities of interest. It supports "local optimization“, that is the information is optimized for the interest of the particular group, not the whole.
- Supports dynamic and accurate dissemination of information. By relying on user generated tags, information and knowledge created on the web reaches the network of the interested participants more dynamically and accurately than by relying on brittle manually entered user profiles which often use irrelevant key word schemes and quickly fall out of date.
- Enables Bottom-up collaboration and Information search. ‘Dynamic folksnonomies’ arise from the group tagging patterns and ‘Google mashups’ are web or desktop widgets that marry up disparate data sources with disparate presentation layers. Main examples of this on the web are GoogleEarth visualizations of nearby places of interest from sites not originally designed for Google, but it could apply to other platforms like mobile handhelds or i-phones.
- Finally, ‘popular social networking’ sites connect people across the net and as a result their popularity has been growing rapidly. There are numerous articles in the popular press about GPS- and wireless-based technologies for keeping track of exactly where your friends are and what they've been doing at any given moment via brief text-blog entries.

While it is the "young and the geeky" that are mainly driving this market, the intelligence community has already adopted Web 2.0 technologies and the operators in the AOC centers often use chat to collaborate. A more concerted effort from the part of the government to

leverage these technologies has the potential to yield great results in systems integration, not so much to automate the exchange of data, but rather to make data available in formats that can be readily analyzed by the users.

5.5 STANDARDS

Central to the interoperability effort is the use of standards, both in terms of expressing the data and also processes to enable applications to work with each other. There are several efforts in both the industry and the government to define standards and achieve adoption from the community. In this effort, we have evaluated two such efforts, the OMG suite of standards and the eXtended Metadata Registry (XMDR).

5.5.1 Object Management Group (OMG) Standards

OMG's CORBA specification established standards for component to component interoperability in a network. However, this standard addresses only one level of granularity for interoperability, and over the years OMG has extended their efforts to address different levels of granularity for interoperability. OMG aims to build consensus in the form of standards for interoperability over heterogeneous systems and networks using modeling as an enabling technology to support the design implementation and maintenance of interoperable systems. The foundation for modeling is OMG's Universal Modeling Language (UML) standard. In the following sections we will list a subset of OMG standards that are related to interoperability. The goal of interoperability can be generalized to mean capturing structured information so that it becomes possible to convey or transmit that information to a different entity, component or toolset.

Many of these standards are ostensibly designed to meet business requirements, but have obvious applicability to government. In many cases, the commercial and government sectors have overlapping or competing standards. One such example is The Open Group Application Framework (ToGAF) and the DoD Application Framework (DoDAF).

OMG's Business Specifications

These include the following:

- Business Motivation Model (BMM)
- Business Process Maturity Model (BPMM)
- Business Process Modeling Notation (BPMN)
- Business Process Definition Metamodel (BPDM)

A Business Process (BP) is a defined sequence of steps to be executed in order to generate a product or a service. Essentially everything a business does is part of a BP. Each BP can be subject to *Business Rules* and can contain multiple *Workflows*. The goal of these OMG standards is to enable Business process modeling. The BPMM is analogous to the Capability Maturity Model Integrated (CMMI), except it is focused on business processes. Like CMMI a BPMM has 5 levels:

- Initial
- Managed
- Standardized

- Predictable
- Innovating

Business process modeling can be used to guide business process improvement, to assess the risks in developing and deploying enterprise applications, to evaluate the capabilities of suppliers, and to compare against the competition. BPMN standardizes flowchart diagrams that are done by business analysts and managers. It can be used by humans for analysis, or it can be mapped to an executable language like Business Process Execution Language for Web Services (BPEL4WS). It can also be used to communicate the structure of business processes internally, or across businesses (B2B).

The Modeling Foundation

Models can represent with clarity and stability processes and applications in a technology independent manner. The help maximize IT return on investment. OMG supports modeling with the following standards (among others):

- MOF: Meta-Object facility
- UML: Unified Modeling Language
- XMI: XML Metadata Interchange
- CWM: Common Warehouse Metamodel
- QVT: Query View Transform

MOF is a self-describing facility that can be used to define modeling languages like UML. XMI is XML based and can be used as a vendor and platform independent language for exchanging both models and meta-models. CWM can be used for Data Warehousing integration and includes record and table formats and specifications for data loading and transformations. QVT is a specification for platform and vendor independent transformations of models.

Prior to MOF and UML 2.0 metadata about objects in the enterprise could be stored in a Naming Service, a Trading Service or an Interface Repository. MOF and UML 2.0 include infrastructure that can be used to define a metadata foundation for modeling.

MOF is a key element in OMG's Model Driven Architecture (MDA) initiative because it can be used to generate code from a model. QVT can be used to define automated transformations that can enhance the utility of the models, and can support model evolution. MOF-based models are technology agnostic. For example a UML data model can be transformed via a UML-to-Relational transformation to a relational database schema. The same model can be transformed via a UML-to-EJB transformation to an EJB Application. Transformations can be specified across different heterogeneous technologies, for example: UML package corresponds to Relational Schema and a Java jar file. A UML class corresponds to a Relational table and a Java class. A UML attribute corresponds to a column of a table and a field of a Java class. The canonical form of a transformation of a source model is to loop through the elements of the model while traversing associations and applying transformations to each element reached according to its type. Metadata forms the foundation for model transformations. By using MOF to standardize the format of the metadata we enable transformation specification and application.

Common Warehouse Metamodel (CWM)

CWM addresses the integration problem across multiple databases, repositories and schemas. Typically data integration requires manual schema transformations. CWM is aimed at integrating

existing data models, can be mapped to existing database schemas, and can support automated schema generation. It can be used to facilitate data mining and OLAP.

CWM consists of a set of metamodels consisting of the following:

- CWM foundation
- Relational
- Record data
- Multidimensional data
- XML data
- Transformations
- OLAP
- Data mining
- visualization
- Business terminology
- warehouse processes
- warehouse operations

CWM is about to be renamed. The new name is Information Management Metamodel (IMM). MOF will be the metamodel for IMM and UML will provide profiles for Relational, Entity-Relationship and XML data modeling.

5.5.2 eXtended MetaData Registry (XMDR) Project

This project is aimed at improving the standards and technology for representing the semantics, terminologies and structures of heterogeneous data sources. This project includes a live prototype implementation of a metadata registry and is also aimed at improving or refining the technologies used to implement metadata registries. Another goal of the project is to propose extensions to the ISO/IEC metadata registry standards for enhanced diversity of metadata types, semantic specification and queries.

The project is spearheaded by a group at the Lawrence Berkeley National Laboratory. Its institutional participants include the Department of Defense (DoD), Environmental Protection Agency (EPA), Department of Energy (DOE), National Cancer Institute (NCI), US Geological Survey (USGS), National Biological Information Infrastructure (NBII) and the Mayo Clinic.

XMDR Use Cases

The MDR can be used by a stand-alone application to manage a portion of the semantic context, classification schemes or data types used by or produced by the application. In the case of multiple interoperating systems the MDR can foster semantic consistency across multiple applications. Data migration across two systems can be facilitated if both systems have registered its metadata with the MDR. Likewise, Data Portals or Warehouses providing access to data from multiple sources presented in a form that has well-defined semantics.

An important use case is support for Data Grid or Online Transaction Networks such as travel reservation networks, securities trading, electronic banking etc. Data grids are also pursued within various scientific communities.

Finally, the most challenging use case is support for a “universal” grid which is not limited to a particular domain or scientific area and contains resources that can be dynamically discovered,

navigated, queried and utilized both by humans and automated software agents and client systems.

A different class of use cases includes repository administration and maintenance. Such activities include registration of metadata, location and retrieval of metadata, revision and extension (while preserving the integrity of the system as a whole), semantic normalization, disambiguation and harmonization.

Additional use cases include mapping between metadata across different domains, relationship maintenance between metadata, navigation, aggregation, resource discovery and online help.

Project resources

A live prototype implementation of XMDR is available at http://xmdr.org. This includes SPARQL query capability and Advanced Inference Search. Project software is available for download. The implementation is also based on bare XML rather than RDF, XMI or SCL.

Architecture

The SOA for XMDR is implemented in a REST (Representational State Transfer see http://rest.blueoxen.net/cgi-bin/wiki.pl?RestArchitecturalStyle) interface that will be later wrapped in SOAP.

The XDMR team has selected an internal architecture for the project based on the so called "fully modular approach" leveraging open source software such as Apache Web Server, Eclipse IDE, and the Protégé Ontology Editor. Each component of XDMR is installed as a "plug-in" for the framework, resulting in a flexible system that is highly reusable and portable.

Areas that the project is not planning to immediately address are: transaction management, classification (in DL sense), and an integrated query language.

Modules (plug-ins) that are proposed for immediate implementation include: Registry Store (persistence and versioning), Metadata Validator, Retrieval Index, Mapping Engine and Authentication Service.

5.6 ROADMAP FOR INFORMATION SERVICES

Our investigation of Information Services technologies has revealed a number of areas in which further progress can be made. In this section, we make a series of specific recommendations for how to further research in these areas.

Recommendation #6: Embrace process modeling

The MDA initiative from OMG is based on the idea that "active" models such as MOF-based UML diagrams or BPDM business process models are intrinsically more valuable than passive models such as charts, viewgraphs or Visio diagrams. The purpose of this effort is to expand this notion to not only work products, but also systems, processes and applications. We could potentially capture AF operating procedures, such as the kill chain, into MOF-based models and use the semantically rich capabilities of the OMG models to enable integration between the various collaborating systems.

Recommendation #7: Establish Semantic Technologies Challenge

There is a wide criticism within the technical and user community that the Semantic Technologies have not solved any real problems yet. In this effort, we will identify a series of hard operational problem and conduct competition of technologies to solve them. Some problems or interest can include Close Air support, Time Critical Targeting, etc. This effort will be similar to the DARPA Grand Challenge and the KDD challenge that the intelligence community is conducting.

The focus of this effort will be to select the best technologies that can solve real operational problems. It will provide a uniform standard for comparing technologies and it will facilitate the selection of the best approaches.

6 RELATED PROGRAMS

In addition to technologies, we also investigated programs related to Semantic Integration in an effort to identify trends and best practices that could be leveraged by a future research effort in this field. In particular, we explored the following programs: Metadata Extraction and Tagging Services (METS), Joint Metadata Tagging Pathfinder, IC Service-Oriented Architecture, BlackBook, TANGRAM, Net Enabled Command Capability (NECC), Future Combat System (FCS) and the GIG.

We observed that the adoption of semantic technologies varies widely by the maturity of these programs. At the lowest end of adoption are programs that have clear path to transition or are ready to be fielded. These programs make wide use of middleware technologies, such as ESB, CORBA, etc. and employ relational or XML technologies for expressing their data models. In addition, we observed that in many of these programs there is a wide plethora of unstructured reports that require processing therefore there is a high need for content tagging and cataloguing.

At the next level up are the Emerging Systems that are coming out of ACTDs, DISA, the Services, etc. These programs have evolved beyond traditional middleware technologies and have embraced Web Services. In fact, there is a mandate by the main sponsors (i.e. DISA, ONR, ARL, etc) for all new services to be net-centric and be implemented as Web-Services. The interoperability in these programs is addressed from the grounds up, mainly through three approaches: (a) mandate a common schema (FCS); (b) use upper ontologies that capture the common terms of all the external systems (NIEM, CBRNE, etc); and (c) develop peer to peer mediators that convert the data to the different specifications.

Finally, at the top of the adoption curve are the Research Efforts, coming out of DARPA, AFRL, DTO, etc, that have embraced Semantic Technologies, such as developing expressive ontologies written in OWL, using machine learning to make inferences and mediate data between different specifications, enrich data requests with context relevant information, etc. In addition to data interoperability, these programs explore the use of planning and composition technologies to build dynamic and adapting workflows of services that emerge and evolve depending on the user request and the needs of the particular mission. Finally, they employ Web 2.0 types of technologies for tagging content, facilitating collaboration among users and promoting social networking.

6.1 METADATA EXTRACTION AND TAGGING SERVICES (METS)

METS is an ontology-based tagging service for documents. The METS services extract metadata information from textual documents, stores the meta-data into RDF and XML representations, and makes this information available to downstream applications for intelligence analysis. The emphasis of the METS program is on generation of large volumes of semantic information, which will drive the development of more sophisticated search and analysis tools using that information.

METS is focused on extracting and disseminating meta-data information from text document. The architecture consists of a Database server tier that stores the metadata in XML and RDF, the Extraction Tier Server that performs the data cleansing, normalization and relationships extraction, and the Client Tier Server that exposes the application of METS as Web enabled

services. The database server uses Tamino for the storage of the metadata, the extraction server utilizes, NetKernel, AeroText, Attensity, Mohomine, ContentMaster, Name Hunter, Name Parser and Name Classifier, for entity extraction, and the client server is deployed on BEW WebLogic.

METS addresses the interoperability problem by using an upper ontology to represent the meta-data information and by storing the extracted meta-data into an RDF repository that enables schema evolution and allows client applications to perform further reasoning and inferencing with the information.

6.2 JOINT METADATA TAGGING PATHFINDER [1], [2]

Another program related to metadata extraction is the Joint Metadata tagging Pathfinder, a Joint effort by the Air Force, Army, Navy, Marines and JFCOM. The main hypothesis of the Pathfinder program is that manual creation of meta-data is inaccurate and costly. According to statistics developed by the A4 cell of the AF to understand magnitude of cost and effort to create the DDMS tags to populate a Metadata Repository, the effort is quantified as follows:

- There are currently 139 high-profile A4 systems that provide reports
- Each system generates about 150 reports on average
- Labor hour to enter 1 document into the metadata registry is 6 hours. This estimate is based on real-life exercise.
- The hourly labor hour rate is $83.

Total estimated cost that would be incurred by the A4/7 cell for populating the MDR with all system reports is $10.4M

The objective of the Pathfinder program is to demonstrate that COTS search engines have the basic ability to automatically identify the content discovery keywords as required by the DoD Discovery Metadata Specification (DDMS).

The Pathfinder program ran experiments with three tools, Autonomy, Convera and FAST and compared the recall performance of the extracts meta-data across four input sources: IL, JC3IEDM, FM and Air Ops. The performance results were mixed. The maximum recall achieved was 93.6% and the lowest one 8.9%. Our source document didn't identify the combination of vendor/data source that yielded these results.

A secondary learning from this program is that instead of attempting to align individual data sources, a better approach will be to align the shared-vocabulary, or upper ontologies, of the various COIs. PathFinder introduces the concept Metadata Environment which is an architecture to achieve integration between COIs. Each MDE Each MDE consists of three registries: (a) structural registry that holds the format and semantics of the data; (b) service registry that describes/advertises services; and (c) metadata catalog that contains descriptive information to enable the discovery of the data.

6.3 INTELLIGENCE COMMUNITY SERVICES ORIENTED ARCHITECTURE (ICSOA) [3]

ICSOA is an integration framework for the Intelligence Community that enables the sharing of data and the interoperability of applications in support of the business needs and the mission

of the IC. ICSOA consists of a set of core services to process and disseminate data (similar to the NCES model), is augmented by a governance process for adherence to the specifications of the framework, and is specifically focused on the 'need-to-share' paradigm set forth by the *National Intelligence Strategy*.

The focus areas of the IC SOA include:

1. IC Data Services – An IC data access platform that enables distributed discovery, access, and consumption of data of relevance across organizational, agency, intelligence discipline, and Community of Interest (COI) lines, regardless of physical location, data type, and/or technical implementation.
2. The definition and development of key service-oriented infrastructure services to enable IC-wide consumer-provider interactions. These service capabilities include Security, Service Discovery, Mediation, Messaging, Management, and Governance.
3. Governance – Define and develop the necessary Architecture Management Board (AMB)-supporting governance constructs to support the AMB to enable IC SOA implementation
4. Assessment, Validation, and Refinement – The IC SOA emphasizes continuous improvement and refinement of architecture specifications through assessment, validation, feedback, and alignment activities. This includes the creation of an environment for interoperability testing using the IC SOA Infrastructure Services and Specifications to:
    - Leverage and validate IC SOA Reference Architectures and specifications for interoperability and data exchange across the IC in order to provide IC SOA implementation policy.
    - Accelerate the adoption and deployment of the IC SOA implementation policy (in existing and new programs).
    - Ensure the architecture can be supported by multiple vendors (COTS-centric).
    - Encourage the widest participation of programs and industry base (integrators).

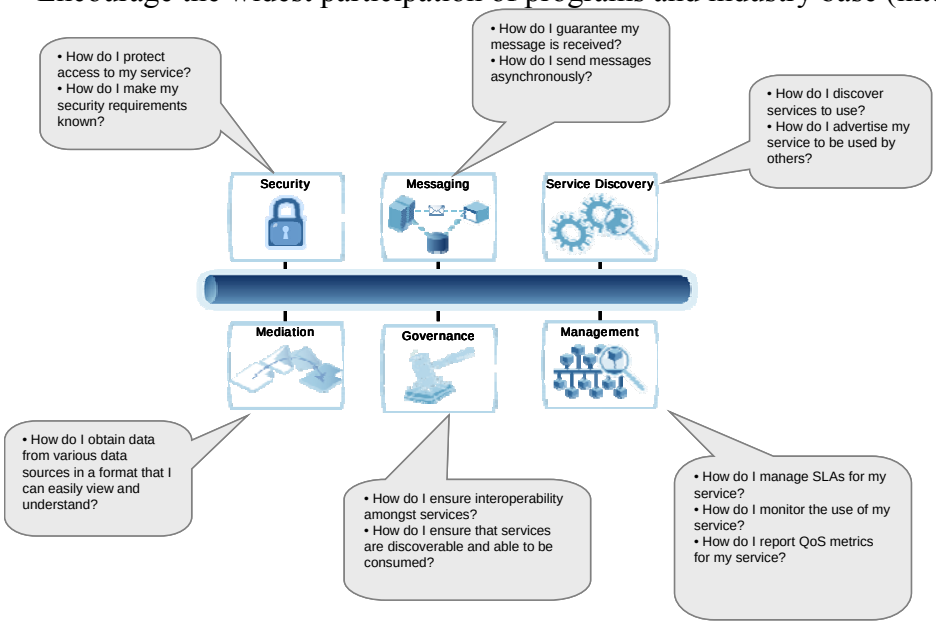


Figure 22: ICSOA Core set of services

The ICSOA consists of 6 core services, as shown in the above figure. Each service addresses specific requirements, as set forth by the ICSOA architecture team, and is implemented via a combination of COTS, GOTS and customer tools.

To develop this infrastructure, the ICSOA program identified a number of potential vendor products to be used for the Reference Implementation and established evaluation criteria to ranks these products. In addition, the team developed a series of pilot case studies that will be used to validate the Reference Architecture and the efficacy of the chosen products to satisfy the needs of the IC for data exchange across the community. So far the team has achieved basic integration of searchable data services via COTS tools, and has integrate over 15 data sources on the RDEC network through a combination of COTS and government related programs.

6.4 BLACKBOOK

Blackbook is a research program from the intelligence community to retrieve targeted information from multiple, heterogeneous data sources. The technical approach of Blackbook is based on the concept of the 'data dip': dip an object of interest in various sources and collect the relevant information only. The actual implementation of the 'data dip' is done via procedurally defined mediators cognizant of the schema and semantics of each external source that translate the data from the source schema to the specification of the search object.

Central to the Blackbook system is a data model that is domain and ontology neutral. The model is simple and consists of a few core entities, such as Nodes, Attributes, Entities, Associations, etc. Domain specific models are expressed in terms of these few primitive structures. This modeling approach is very similar to graph based models that are applicable for Intelligence related applications. Data that have been retrieved from the external sources and expressed in the BB model can be processed by BB client applications. BB provides an object-oriented approach for external developers to define BB compliant apps. Apps are graph based and their signature model is very simple:

```
Public BBGraph (BBGraph input){
// do work
Return BBGraph
}
```

The vision behind Blackbook is to provide knowledge collaboration between individual analysts, COIs or even whole departments and agencies, operate over many massive databases, be fully distributed, support data normalization for fast access, support policy filters for need-to-know access, deploy in operational environment and also serve as a venue for further research in the area of data integration. The next version of BB (v 2.1) will use RDF, though in a somewhat restrictive manner, as a central storage representation

6.5 NET ENABLED COMMAND AND CONTROL (NECC) [4]

NECC is a major effort by DISA to establish a common joint standard for a GIG Computing Node (GCN) physical architecture. The standard is based on the use of VMWare as a virtual environment to establish a common computing platform across different hardware servers and software operating systems. Applications that are certified to run on this common platform can be deployed easily on the various nodes of the NECC framework depending on the availability of the resources and the needs of the mission. This approach has several benefits: (a) because of

the commonality of the computing platform, applications for the NECC Grid Computing Node (GCN) are easier to deploy and operate, thus resulting in significant cost savings; (b) the footprint of NECC compliant systems are smaller than fully deployable applications, thus eliminating the need of large hardware systems and allowing applications to run on resource-constraint server rooms (i.e. Navy ships); (c) because of the well-specified computing configuration and the disciplined governance of building the infrastructure, software is more readily provisioned thus resulting in new capabilities delivered in more rapid pace; and (d) software is more easily accredited, since the accreditation on a single VM platform makes the software valid across all GCNs.

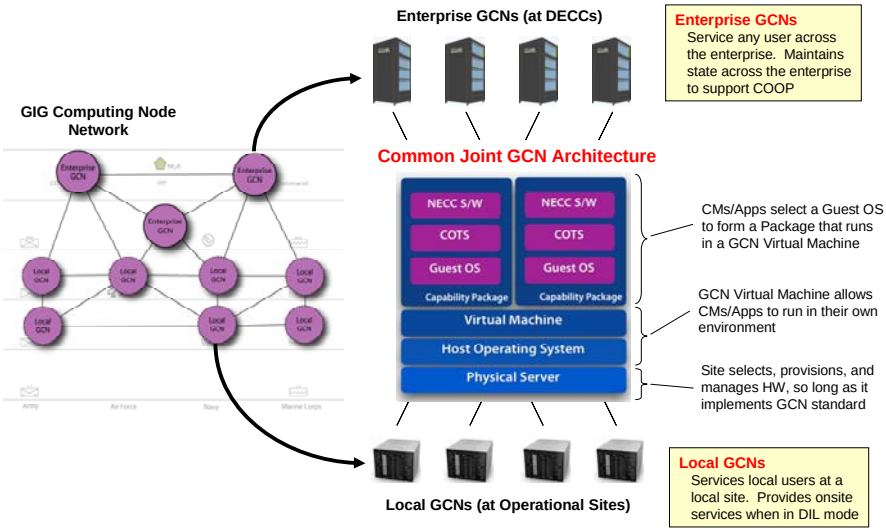


Figure 23: NECC specifies a common computing platform for the Grid Computing Nodes

As Figure 23 shows, the NECC specification supports two types of nodes: Enterprise GCNs that service users across the whole enterprise, and Local GCNs, that service users that are local to a specific site. Both Enterprise CCNs and Local GCNs operate on top of the same virtual computing environment, thus making them interchangeable in terms of where to deploy, and ensuring transparency of access from the client applications.

In addition to defining a common computing platform, NECC will also deliver a set of core services and a specification for third-party developed services. The core NECC services consist of (a) Core Enterprise Services that includes Security, Data Discovery, M2M messaging (preferably Enterprise Service Bus), collaboration (provided by NCES); (b) Redirection Services that allows a service request to automatically redirected to a physical service access point (utilize a COTS solution); (c) Geospatial Rendering services supporting many different formats and processing capabilities for geospatial data; and (d) Analysis and Reporting services such as tables, charts, etc.

The third-party services fall into three major categories: (a) presentation services for visualization and UI capabilities (for example, web based mapping in the Global UDOP apps; (b) synchronous data access, for query-able and discoverable data capabilities (for example, unit readiness report lookup in the Force Readiness app); and (c) asynchronous data provision service that sends data updates to subscribers (for example, publishing track updates in the Joint Track Data apps.

NECC is in the first of many increment spirals. Increment #1 has identified a set of data sources to integrate and processing applications to use and is scheduled for completion in August 07. The evaluation of increment #1 will be conducted on a testbed consisting of 6 computing nodes operating over the SIRPNET.

6.6 THE EXPANDING GLOBAL INFORMATION GRID (GIG)

Over the last decade, the United States Department of Defense and Armed Services have committed to the establishment of a networked force to fully leverage the power of information as a combat multiplier. The manifestation of this effort has come to be called the Global Information Grid, or GIG. Joint Pub 6-0 defines the GIG as the "globally interconnected, end-to-end set of information capabilities, associated processes and personnel for acquiring, processing, storing, transporting, controlling, and presenting information on demand to joint forces and support personnel." The enabling technologies which constitute the GIG provide for the Joint services to operate in a more integrated manner with each other as well as with the broader communities of interest within and external to the US Government.

The challenge facing the Air Force is to identify, define, and develop the information management services which will enable its Air Operations Centers, Air Wings, and Bases to operate effectively within the larger GIG, and more importantly, take advantage of the ever increasing amount of information available on the GIG.

Before presenting the challenge facing the Air Force, it is important to further describe the GIG in order to identify which elements require investment to resolve challenges with respect to Air Force Battle Management Systems. The GIG consists of seven components: Warrior, Global Applications, Computing, Communications, Foundation, Information Management, and Network Operations. Figure 24 demonstrates and describes the relationship of each of these components to one another.

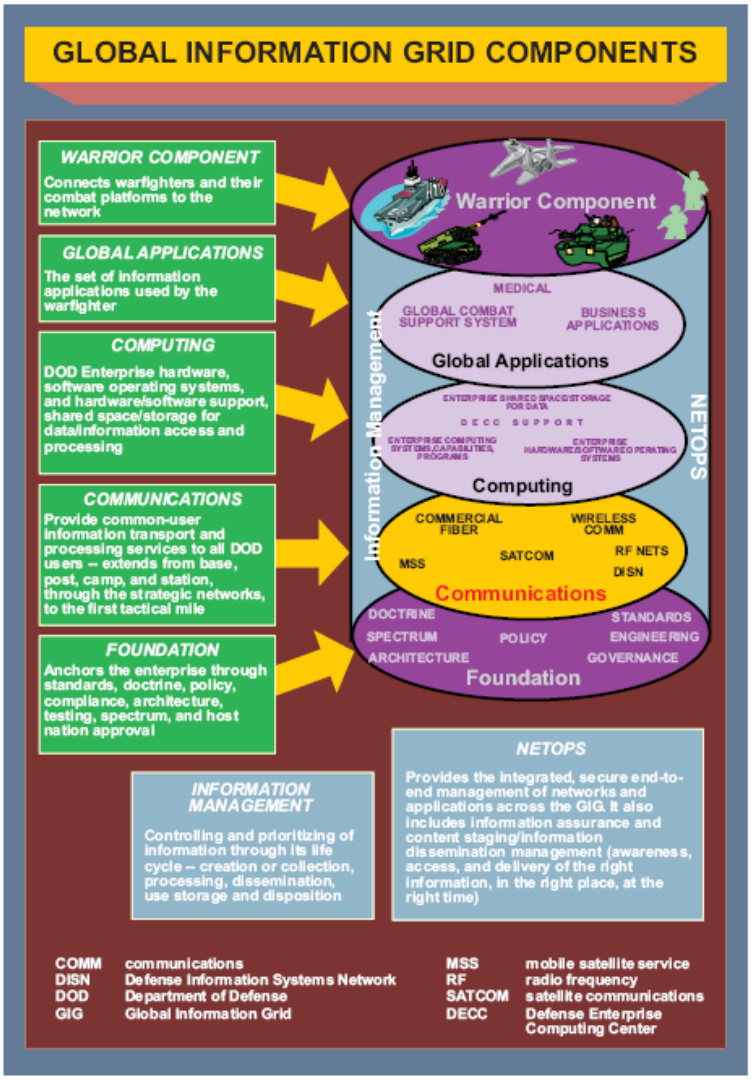


Figure 24: GIG Components (Reference JP 6-0)

As can be seen Figure 24, these are two components that serve to integrate the other five components. These are Network Operations and Information Management. The Network Operations Component provides the mechanisms to manage the GIG, while the Information Management Component provides the Information Distribution and Transformation Services required to deliver the correct information to the correct warfighter when required. Figure 25 further describes the services which comprise the Information Management Component.

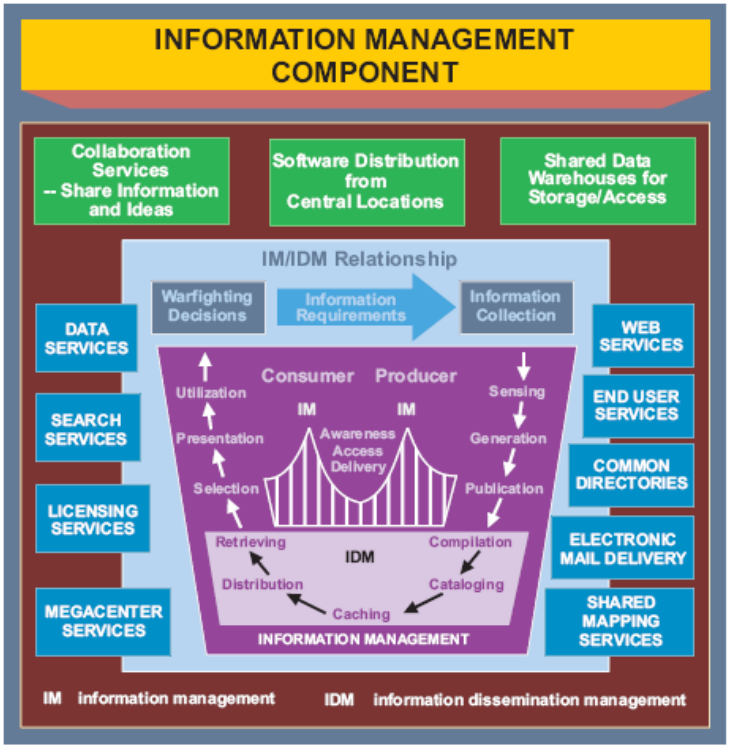


Figure 25: Information Management Component (Reference JP 6-0)

The Information Management Component handles information discovery, sharing, mediation, and delivery to the appropriate warfighter in a manner which facilitates situational awareness and decision making. The challenge here is having Information Management technologies keep pace with ever growing global communications, data storage sources, transmission speeds, and information availability. To date, no common data standard exists for finding, accessing, and storing information or for specifying the semantics associated with the data. Producers create, publish, and store information in a format consistent with their known consumers; however, the GIG environment provides for the reduction of these traditional stovepipes such that information can be used by anyone who may require it. Yet, the inconsistencies in this format create a challenge for the "non-traditional" consumer. Additionally, the consumer is challenged by the fact that information systems today are incapable of understanding the context of the analytical task the information is required for, which limits the communicative efficiencies of human-machine interaction.

While the GIG requirements do require the establishment of standards for sharing data, the current standards still have shortcomings; and it is unlikely that previously published information or non-GIG managed information sources will migrate to these standards. This suggests a need for technological investment in technologies to assist in the discovery, mediation, and transformation of the information provided by these sources.

6.7 U.S. ARMY FUTURE COMBAT SYSTEMS NETWORK

The Army's Future Combat Systems (FCS) program is delivering a System-of-Systems consisting of soldier equipment and 14 platforms all connected via the network (commonly referred to as 14 + 1 + 1). As shown in the figure below, the FCS Brigade Combat Team nests

its network inside the Army's LANDWARNET, which is the Army component of the Global Information Grid (GIG).

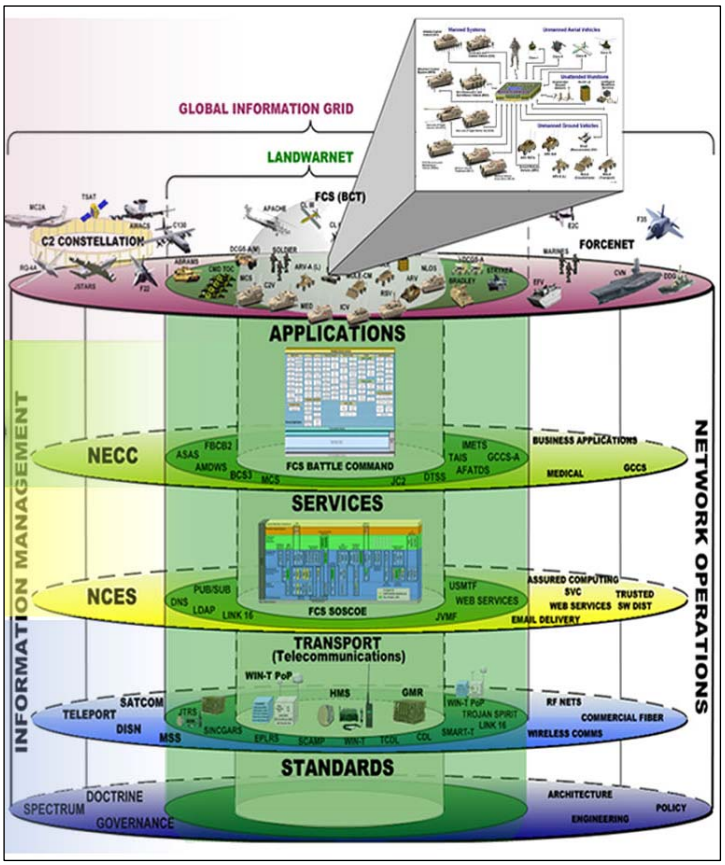


Figure 26: FCS BCT as part of the GIG

The FCS Brigade is the ground force component of the GIG where the primary transport layer is provided by the Joint Tactical Radio System (JTRS) and WIN-T. The FCS program is developing a System-of-Systems Common Operating Environment (commonly called SoSCOE) to serve as the Information Services and Interoperability Layer. In the tactical environment, SoSCOE is expected to provide service discovery of available sensors and effectors to respond to battlefield events.

SoSCOE delivers Net-Centric capabilities through 16 service families which are grouped into 4 Integration Groups: Tactical Integration, Platform Integration, Battle Command Integration, and Enterprise/GIG Integration.

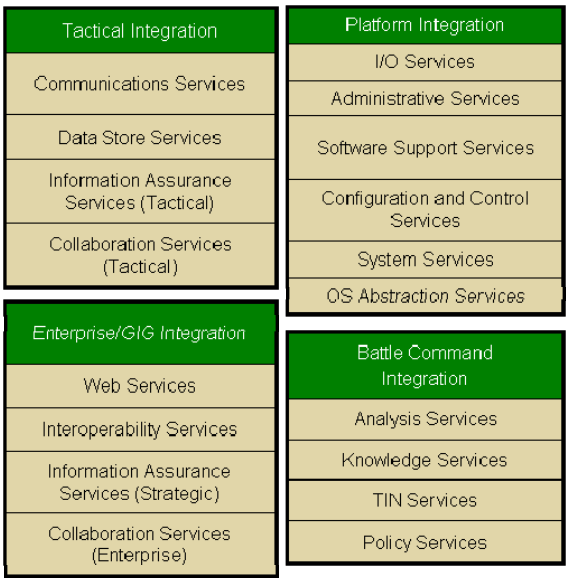


Figure 27: SoSCOE Services and Integration Groups

Tactical Integration delivers interoperability within the FCS Brigade Combat Team elements to include management of network quality of service in a limited bandwidth environment. Battle Command Integration provides a framework for Battle Command Applications. Platform Integration provides basic services common on all platforms. Finally, Enterprise/GIG Integration provides access out to other service providers such as the Net Centric Enterprise Services (NCES).

7 TECHNICAL PROGRAM SUMMARY

The ultimate goal of this trade study is to provide a roadmap with recommendations for SI research and development in future years to help overcome critical limitations in current SI capabilities. This section summarizes our accomplishments in performing the trade study and defining the roadmap.

In our trade study of semantic mediation technologies, we have

1. *Developed an overarching challenge problem scenario* that demonstrates the need for SI technologies. The challenge problem scenario also illustrates how candidate SI technologies might be applied to improve Air Force net-centric operations.
2. *Investigated current and emerging technologies that address semantic alignment*, including information representation and mediation (information transformation). We have looked at technologies applicable to a wide vertical spectrum of information richness, from database schema matching to ontology mediation.
3. *Investigated emerging technology solutions and open standards and architectures for semantic information services* spanning the four abstraction layers of database, object, syntactic, and semantic-level services. Our investigation focused in particular on solutions that are compatible with DISA's Core Enterprise Services Strategy for the GIG.[1]
4. *Evaluated the SI alignment technologies and information services* identified in the two previous tasks, based on an objective set of criteria, including general strengths and weaknesses, comparison of published results, maturity (i.e. Technical Readiness Level), automation support, scalability, and compatibility and applicability to the GIG.
5. Finally, we *specified a roadmap for future SI research and development* that identifies areas where current technologies and services fall short of AF and GIG interoperability requirements and suggest specific technical foci for future research to fill these gaps. In particular we proposed the following 7 research efforts
    i. Develop a Formal Theory of Matching
    ii. Perform rigorous evaluations of semantic alignment technologies
    iii. Exploit rich information for alignment
    iv. Place emphasis on user-assisted semantic alignment, not on complete automation
    v. Develop an explanation capability for matchers
    vi. Embrace process modeling
    vii. Establish Semantic Technologies Challenge

[1] http://www.defenselink.mil/nii/org/cio/doc/GIG_ES_Core_Enterprise_Services_Strategy_V1-1a.pdf

8 APPENDIX A: EVALUATION CRITERIA

Tool: COMA++
Developer: University of Leipzig
Function: Find a mapping between a source ontology or schema and a target ontology or schema.
Features: COMA++ is a composite matcher that provides flexible rules for combining the outputs of multiple schema matchers. It works with relational database schemas, XML schemas, and OWL ontologies. It allows for user input into matching and can make use of past matches.
Evaluation:

- **General strengths and weaknesses:** COMA++ is state of the art for schema matchers, as contrasted with more recent ontology matchers. It permits great flexibility in the way matching is done and has a large number of built-in matching algorithms. It shares a common drawback with other schema matchers, namely that it does not make use of context information in matching (information about the domain or use of a data source or schema). It does not provide any estimate of the degree of confidence in a match. The similarity values it outputs for paired items are hard to interpret.
- **Published performance results:** COMA++ achieved high recall and precision scores on the test cases for the 2006 Ontology Matching Initiative. It was not the highest scoring matcher, however. It should be kept in mind that COMA++ was originally designed as a schema matcher, not an ontology matcher.
- **Maturity:** High. COMA++ is the result of many years of research on schema matching and incorporates techniques from previous well-known schema matchers.
- **Degree of automation support**: COMA++ is fully automated but allows for user input.
- **Scalability**: COMA++ does not scale to very large datasets. Large datasets must be manually partitioned in order for COMA++ to process them.
- **Compatibility and applicability to the GIG specification and reference implementations:** COMA++ was not designed with reference to the GIG; however, it is written in pure JAVA and should be portable to a GIG environment.

Tool: Automatch
Developer: George Mason University
Function: Find a mapping between a relational source schema and a relational target schema.
Features: Automatch is a single algorithm matcher that that uses naive Bayes learning on instance data to evaluate similarity of schema elements. Schema information is not used.
Evaluation:

- **General strengths and weaknesses:** Automatch's strength is a principled probabilistic foundation for part of the matching process. However, this is combined with some ad hoc procedures for computing an optimal match.
- **Published performance results:** Published results are for very small schemas. Precision and recall are in the low 80% range.
- **Maturity:** Questionable. Cannot discover recent work on Automatch. May not be maintained.
- **Degree of automation support**: Fully automated. No user input.

- **Scalability**: In doubt. There is no data on Automatch's performance on large datasets.
- **Compatibility and applicability to the GIG specification and reference implementations:** No information.

Tool: LSD/GLUE
Developer: University of Washington
Function: Find a mapping between a source schema and target schema.
Features: LSD matches new data sources to a previously determined global schema; GLUE extends LSD by matching between arbitrary data sources. Both are composite matchers. Individual matchers use a machine learning approach and their results are combined. LSD and GLUE learn from previous matches and can accept user input.
Evaluation:

- **General strengths and weaknesses:** Good performance on real datasets. One disadvantage is that learning requires manual creation of matches.
- **Published performance results:** Precision and recall are around 80% for real world data. GLUE achieved high accuracy on very large datasets.
- **Maturity:** Good. Continuing development of capabilities.
- **Degree of automation support**: Fully automated. User input accepted
- **Scalability**: Scales to large datasets.
- **Compatibility and applicability to the GIG specification and reference implementations:** No information.

Tool: iMAP
Developer: University of Washington and University of Illinois
Function: Find a mapping between a source schema and target schema.
Features: For each source schema element, iMAP search the space of all matches, returns a set of candidate matches, and uses LSD to evaluate them. iMAP can return *complex* matches, where a term in one schema may map to a combination of terms in the other schema. (E.g. "address" might map to a combination of "city" and "zip code."
Evaluation:

- **General strengths and weaknesses:** Has achieved high accuracy (92%) in real world domains with up to 40 elements. A drawback is that it requires fairly active human in the loop interaction
- **Published performance results:** 43% to 93% accuracy for real world data. High accuracy requires human intervention.
- **Maturity:** Good. Continuing development of capabilities.
- **Degree of automation support**: High accuracy achieved only with human intervention.
- **Scalability**: No data on large datasets, but scalability is in doubt because of the size of the search space.
- **Compatibility and applicability to the GIG specification and reference implementations:** No information.

Tool: Falcon
Developer: Southeast University, Nanjing, China

Function: Find a mapping between two ontologies.
Features: It combines the output of a linguistic similarity matcher and a structural matcher and uses rules to determine how much weight to give to each matcher for a given match problem.
Evaluation:

- **General strengths and weaknesses:** Does very well when structural similarity between the two ontologies is high, even when linguistic similarity is low; does not perform so well when structural similarity is low.
- **Published performance results:** Had best average performance in 2005 Ontology Evaluation Initiative.
- **Maturity:** Was developed fairly recently but builds on well-known technologies.
- **Degree of automation support**: Fully automated; no user input.
- **Scalability**: Has gotten high precision and recall on test sets with up to 10k classes and several dozen relationships.
- **Compatibility and applicability to the GIG specification and reference implementations:** No information.

9 REFERENCES

[1] 'Joint Metadata Tagging Pathfinder', Presentation to DON CIO, Sep 11, 06

[2] 'USAF Data Strategy: A Federated Approach to Net Centricity', Air Force/Army/Navy/ Marines/JFCOM Team, 11 September 06

[3] 'IC Service-Oriented Architecture', Overview Presentation to the Director of the National Intelligence Office of the CIO, December 2006

[4] 'NECC Technical Review to the C2 PEO', Mark Kuzma, 16 Feb 2007

[5] 'Data Interoperability across the Enterprise – Why Current Technology Can't Achieve it', http://209.85.165.104/search?q=cache:zfYAkTcnl2MJ:colab.cim3.net/forum/cuo-wg/2007-02/docGnfb8BKQyl.doc+%22current+technologies+provide+no+viable+solutions+for+sharing%22&hl=en&ct=clnk&cd=3&gl=us, paper was drafted and reviewed by members of the Cross-Domain Semantic Interoperability (CDSI) Working Group, http://www.visualknowledge.com/wiki/CDSI, and its parent organization, Semantic Interoperability Community of Practice (SICoP), http://colab.cim3.net/cgi-bin/wiki.pl?SICoP . Editor: James Schoening, U.S. Army, james.schoening@us.army.mil

[6] Pioch, N., Barlos, F., Fournelle, C., Stephenson, T., "A Link and Group Analysis Toolkit for Intelligence Analysis", Proc. 2005 International Conference on Intelligence Analysis, McLean, VA, May 2005.

[7] Rosenthal, A. and Seligman L., "Scalability Issues to Data Integration." AFCEA Federated Database Colloquium, San Diego, 2001.

[8] Noy, N, Doan, A., and Halevy, A. Y., "Semantic Integration," *AI Magazine*, Spring 2005, pp. 7-9.

[9] Hunter, D. and Bostwick, D., "Default Reasoning with Contexts." Contexts and Ontologies: Theory, Practice and Applications: Papers from the AAAI Workshop. Technical Report WS-05-01, AAAI Press, Menlo Park, California, 2005

[10] The OWL Services Coalition, "OWL-S: Semantic Markup for Web Services." http://www.daml.org/services/owl-s/1.0/owl-s.pdf

[11] McIllraith, S. and Son, T, "Adapting Golog for Composition of Semantic Web Services." Proceedings of the Eight International Conference on Knowledge Representation and Reasoning, Toulouse, France, 2002.

[12] Wu, D., Parsia, B., Sirin, E., Hendler, J., and Nau, D., "Automating DAML-S web services composition using SHOP2." In *Proceedings of 2nd International Semantic Web Conference (ISWC2003)*, Sanibel Island, Florida, October 2003.

[13] Doan AnHai, Halevy Alon, "Semantic-Integration: Research in the Database Community", American Association for Artificial Intelligence, Spring 2003.

[14] Su, Xiaomeng, Hakkarainen, Sari and Brasethvik, Terje, "Semantic Enrichment for Improving Systems Interoperability." In *Proceedings of the 2004 ACM Symposium on Applied Computing*, March 14-17, 2004, Nicosia, Cyprus.

[15] W3C. Web Services Architecture. http://www.w3.org/TR/ws-arch/

[16] Scott Renner, Arnon Rosenthal, James Scarano, "Data Interoperability: Standardization or Mediation," The Mitre Corporation.

[17] Michalowski, Michael, Thakkar, Snehal, and Knoblock, Craig, "Automatically Utilizing Secondary Sources to Align Information Across Sources." In *AI Magazine*, Spring 2005, pp. 33-44.

[18] Defense Strategic Integrated Decision Environment, http://www.les.disa.mil/c/extranet/home?e_l_id=50

[19] JMS Performance Comparison: Performance for Publish Subscribe Messaging 10/29/2004, Krissoft Solutions for ITtoolbox EAI http://research.ittoolbox.com/white-papers/

[20] Do, Hong-Hai; Melnik, Sergey; Rahm, Erhard. "Comparison of Schema Matching Evaluations" Proc. GI-Workshop "Web and Databases", Erfurt, Oct. 2002.

[21] Euzenat, Jerome, Stuckenschmidt, Heiner, and Yatskevich, Mikalai. "Introduction to the Ontology Alignment Evaluation 2005." In *K-CAP '05, Integrating Ontologies Workshop*, October 2, 2005, Banff, Alberta, Canada.

[22] Do, Hong-Hai and Rahm, Erhard. "COMA – A System for Flexible Combination of Schema Matching Approaches." In *Proceedings of the 28th VLDB Conference*, Hong Kong, China, 2002.

[23] Doan, AnHai. "Learning to Map between Structured Representations of Data." Ph.D. thesis, Computer Science & Engineering Department, University of Washington, 2002.

[24] Cyc, http://www.cyc.com/.

[25] Haarslev, Möller, Ralf, Wessel, Michael. "Description Logic Inference Technology: Lessons Learned in the Trenches." Available at http://www.sts.tu-harburg.de/~r.f.moeller/racer/dl-2005-racer.pdf.

[26] Srin, Evrin; Parsia, Bijan; Grau, Bernardo; Kalyanpur, Aditya; and Yarden, Katz. "Pellet: A Practical OWL-DL Reasoner." Available at http://www.mindswap.org/papers/PelletJWS.pdf.

10 ABOUT THE AUTHORS

Fotis Barlos

Dr. Barlos has 15+ year experience in software development, program management and technology research in the areas of decision support tools, knowledge inference, planning, effects based operations, automated computing, parallel processing and data warehouses. He has managed multi-million dollar programs for both Defense and Commercial systems. Dr. Barlos has a Ph.D. in Computer Science from George Mason University and had published several papers in Data Processing, System Architectures and Performance Analysis. Prior to joining AIT, Dr. Barlos held senior technical positions at Epsilon Data Management and Vignette Corporation, where he led the technical track, and provided project management for several multi-year engagements for large financial, e-commerce, and health-care systems.

Dan Hunter

Dr. Hunter has extensive experience in the areas of ontological engineering, design of representation languages, and translation between different languages. At BAE-AIT (formerly ALPHATECH, Inc.) since June 2001, Dr. Hunter has led ontology development efforts in the areas of terrorist threat detection and detection of complex events in videos. On the VACE program, he was a member of a working tasked to develop a language and upper ontology for describing video events. He has implemented translators between a variety of representation formats, including a translator from DAML+OIL into AKS. As an ontologicial engineer at Cycorp, Dr. Hunter was responsible for the development of ontologies in multiple military domains, including command structure and representation of urban warfare. While at Cycorp, he implemented an interface between the University of Maryland's SHOP planner and the Cyc knowledge-base.

Basil Krikeles

Dr. Basil C. Krikeles is Chief Architect for the Intelligent Systems Division at BAE Systems, Advanced Information Technologies. He holds a B.A. in mathematics from Princeton University and M.S. and Ph.D. degrees in mathematics from Yale University. His professional interests include large-scale software development, software producibility, distributed computing, object-oriented programming, generative programming, model-driven software development, and numeric/scientific computing.

James McDonough

Prior to joining BAE Systems AIT, Mr. McDonough served as a Captain in the U.S. Army form 1997 to 2004. From 2003 to 2004, Mr. McDonough served as the 4th Infantry Division's Fires and Effects Cell Liaison Officer to Combined Joint Task Force-7 (CJTF-7) during Operation Iraqi Freedom. Mr. McDonough's primary duty was to coordinate, plan, and synchronize 4th Infantry Division's lethal and non-lethal effects targeting activities with adjacent and higher headquarters. This included participation in the Joint Targeting Cycle and submission of Air Support Requests for his division. Since joining BAE Systems AIT, Mr. McDonough has provided Subject Matter Expertise and Systems Engineering in the development of the Planning and Preparation Services for the Army's Future Combat Systems Program.