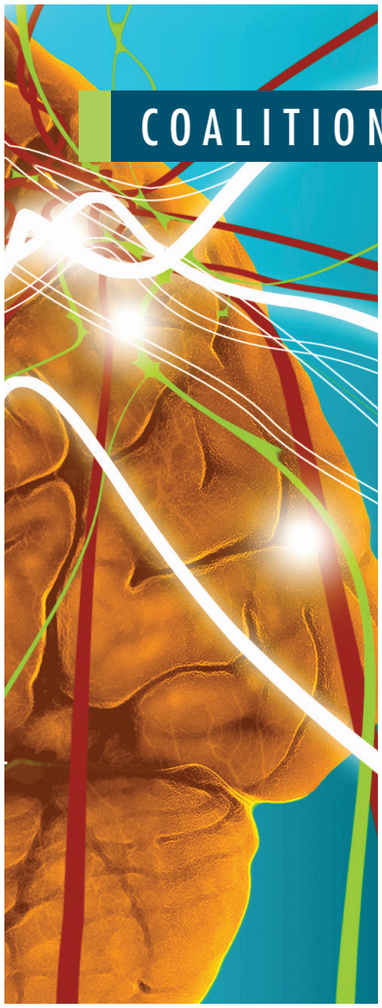

Knowledge-Based Approaches to Information Management in Coalition Environments

Andrzej Uszok, Larry Bunch, and Jeffrey M. Bradshaw, *Florida Institute for Human and Machine Cognition (IHMC)*

Thomas Reichherzer, *University of West Florida*

James Hanna and Albert Frantz, *US Air Force Research Laboratory Information Directorate*

The community of interest information-sharing model lets coalition partners publish and disseminate data in a controlled fashion. In this vein, the authors have extended the Phoenix information management system to improve document selection and filtering.

Information systems are vital to the successful execution of any military or civilian mission. For such systems to be effective in a diverse coalition environment, they must help coalition partners deliver and share information in a timely fashion. *Communities of interest* (COIs) are collaboration models in which different user groups join together to exchange information in support of a common goal or interest.[1] Using a COI lets participants from different groups develop a joint understanding of a mission that will enable them to make decisions independently.

Coalition partners must release information in a controlled manner to avoid leaking private or classified content. They often operate under nondisclosure policies that describe the type of information that must not be released to other coalition partners, and that typically require a human reader to interpret and apply them. Traditionally, nondisclosure policies have been applied through reliable human review (RHR) undertaken by, for instance, foreign disclosure officers. FDOs read a document and mark up information that must be redacted prior to release. With information overload reaching record proportions, however, human analysts are overwhelmed and can't keep up with the sheer volume of data that must circulate among coalition partners. Support tools are required that can assist with the review and

1541-1672/13/$31.00 © 2013 IEEE
Published by the IEEE Computer Society

release process to ensure the timely exchange of critical information.

Here, we describe extensions to Phoenix, an information management system (IMS) from the US Air Force Research Laboratory (AFRL). The Phoenix architecture implements a COI that lets participants

- publish information consistent with policies,
- subscribe to information from coalition partners,
- develop policies that govern information release, and
- provide feedback to the system to improve information selection and filtering.

Our extensions are based on case-based and semantic reasoning, respectively. They use automated methods to model user interests and design and apply policies.

Phoenix Information Management System

Information systems provide a platform for publishing and sharing information across organizations. Phoenix[2] is a service-oriented specification for IMSs. It supports a publish-subscribe-query model for disseminating and accessing information (see Figure 1) and standardizes interfaces for client applications to facilitate client integration.

We've previously built mechanisms that let Phoenix model and provide life-cycle support for COIs.[1] We've also developed a policy-governed[3] Federation Service[4] that lets us securely extend Phoenix across different COIs.

However, Phoenix's static, constrained information format lacks a mechanism for constructing semantic information relationships and other knowledge-based techniques to dynamically correlate and filter information for delivery to clients, preventing it from meeting the needs of complex Air Force missions involving coalition partners.

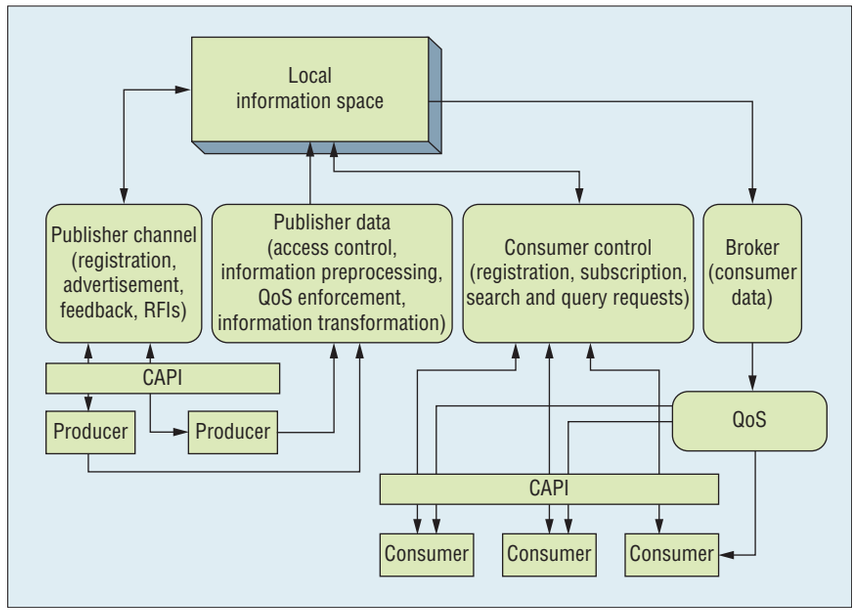

Figure 1. Conceptual architecture of the Phoenix information management system.

We address this limitation with a new approach. The enhanced information model must accommodate different data formats, and the new mechanisms must be able to discover and classify relationships. The approach can then use these relationships to provide dynamic information groups of related information that the system can manage and exploit automatically or with human assistance. This capability enables far more dynamic and meaningful subscriptions and queries for coalition clients.

Information Extraction and Delivery

Here, we look at two approaches for content selection and delivery. The methods are complementary in how they identify content and determine whether it can be released without violating policies and whether it is relevant to a coalition partner's interests.

Text Classification and Information Extraction

Systems that address information-sharing problems must be able to determine information's relevance as it regards certain types or categories and thus distinguish what information is shareable or of interest to others. In addition, such systems need to identify the specific text excerpts in a document that make the content private. Thus, we can view information sharing as a two-fold problem. First, we must identify a specific piece of information within the text—specifically, the information that might or might not be authorized for sharing. Second, for policy implementation purposes, we must determine the category/classification of the text that contains the aforementioned specific information. For instance, does the entire text fall under a specific category, such as "military intelligence"?

To effectively share information, determining only that an entire text document contains private information isn't sufficient. Depending on the recipient, a system might have to point out the specific private content. Our proposed approach addresses the two-fold information-sharing problem by representing security policies as cases in a case base that release information from a higher to a lower security domain. Users can later apply

knowledge that the case base captures to quickly identify information within text documents that's considered private with respect to a security domain and therefore must be removed prior to release.

To deliver relevant information without leaking private content from a document, we've built a sentence-based text classifier that uses marked-up text documents as a training set. The markups represent different user-defined information categories. Once fully trained, the classifier can categorize the content in new text documents and easily determine its relevance to subscribers, who simply state their interest in terms of the categories a given classifier recognizes. Coalition partners with expressed interest in certain information types can forward the information, and can develop and apply category-based release policies.

Given a training set, the knowledge-based classifier breaks marked-up text into sentences and stores each unique sentence along with category information for subsequent classification tasks. When sufficiently trained, the knowledge base contains various examples of content classifications that the classifier can use in the future. Each knowledge base captures classification knowledge for a particular domain. Coalition partners can develop their individual knowledge bases to describe the domain of their authored content. They can also make category labels describing shareable content available to other coalition partners so those partners can learn about the content types available within the coalition.

The classifier uses a text-analytical component that applies natural language processing techniques such as pronoun reference resolutions[5] and shallow-semantic parsing[6,7] to build machine representations from marked-up sentences. The pronoun-resolution step replaces pronouns with their corresponding noun phrases. The shallow-semantic parser breaks the resulting sentences into *constituents* to build a case representation. Constituents are words extracted from the original sentence that describe the different semantic roles that the words have within it. For example, the semantic constituents of a sentence on a sports event might describe activities, actors, and the location where the activities occur. In essence, semantic role parsing attempts to label a given sentence with who did what to whom, when, where, and so forth. After the semantic-role parser identifies a sentence's different constituents, the text classifier constructs a feature vector, with each constituent representing a separate feature. In essence, the vector captures an example of a particular information type at the sentence level that can be used to label semantically similar sentences.

Case-based reasoning (CBR) is a problem-solving methodology that retrieves and adapts previously stored cases to solve similar problems.[8,9] Our prototype text classifier uses a feature vector, compiled from an unmarked sentence, as a problem description that it can then use to choose similar cases from a case library. From the chosen cases, the classifier derives a label to classify the sentence. If no cases are similar, it applies no label, and the sentence is noncategorized. The classifier uses a similarity threshold to decide on the similarity between a problem description and cases in the case base.

The classifier uses a bottom-up approach to examine individual sentences within paragraphs to assess a classification level for the paragraph. For example, *top secret* information supersedes *secret* information in a paragraph classification. Finally, the classifier assigns the entire document a classification label based on the policy and the paragraph classification. Consider a paragraph that contains a sentence labeled according to policy A and a few sentences labeled according to policy B. If policy A characterizes sentences as top secret and policy B characterizes only secret material, then the entire paragraph is classified as top secret.

To select cases from the case base, the CBR component uses a *k*-nearest neighbor algorithm. It measures similarity between the problem description and each case in the case library to select the *k* "closest" cases as candidates for a solution based on a similarity metric. Finally, from the selected cases, the classifier determines what label to assign to a given sentence via majority voting. It chooses the label with the largest frequency among the selected cases to mark the sentence.

The *k*-nearest neighbor selection process uses a similarity metric that measures the distance between a problem description and cases in the case library to select closely matching cases. The intuitive distance between any two cases in the library will consistently reflect the degree of semantic similarity between the two corresponding sentences. For example, consider the following three sentences:

- S_1: John bought a house.
- S_2: John rented an apartment.
- S_3: A truck hit Mary's car.

S_1 and S_2 deal with living accommodations in which John is the actor. S_3 has nothing to do with living accommodations and involves an object as an actor. Intuitively, the semantic distance between S_1 and S_2 is shorter than between either S_1 and S_3 or S_2 and S_3.

To measure sentence similarity, we must break each sentence into its constituents, such as actors, actions, and

objects, for comparison. We assume that two sentences are semantically more similar if their corresponding constituents match. Thus, to measure two sentences' semantic similarity, we break them into the corresponding constituents and compare them word by word. In comparing the different constituents, we might assign them different weights that reflect their importance in a sentence classification. For example, a higher weight on an actor constituent indicates that actors play a more critical role when classifying a sentence. Finally, to compare constituents extracted from two sentences, we analyze their corresponding words and apply Lin's thesaurus[2] to measure word similarity between them.

Lin's between-word similarity measure is based on words' probability distribution within a 22-million word corpus. It considers how often a given pair of words is used in a similar way (for instance, as the subject or the direct object of the same verb). The more relationships we can observe, the higher the similarity value.

Preliminary results have shown that our proposed methods successfully detect and label private content in unstructured text that mustn't be shared. Further work is needed to improve the accuracy of shallow-semantic parsing and the construction of cases for the case library. Also, we need methods to retrain the classifier based on reviewer feedback. For example, reviewers must be able to select an incorrect mark-up the classifier has suggested and assign it the correct classification label or declare the marked-up text as public information for release. User feedback might lead the CBR engine to remove a case that contributed to incorrect conclusions or changes in the weights associated with the feature vector's individual attributes.

In future work, we must extend existing methods to develop scalable solutions. The current techniques require that the classifier compare each sentence to every case in the case base to assess its sensitivity level. The case base itself grows linearly as users feed more marked-up text into the system, thus slowing down the execution of any new markup task. Case-based maintenance[10] addresses performance issues typically associated with continually updating case bases by removing selected cases while preserving case-base competence. Other approaches to consider include hierarchical CBR,[11] which combines both abstract and concrete case representations to solve these problems. We could adapt these approaches to build case representations at different granularity levels; this would let the system identify sensitive content in text by first analyzing entire documents, then paragraphs, and finally sentences to quickly discern private from public content.

Inferring Dynamic Semantic Relationships

To exploit the structure and semantics of managed information and various information type representations, we used an ontology mediation layer.[12] Mapping from a less-expressive representation such as XML Schema to a more-expressive one such as the Web Ontology Language (OWL) is feasible. The ontology mapping is annotated with information about the translated source document's original elements to enable reverse translation. We've developed a document ontology representing a generic document structure and various formats including Microsoft Word and PowerPoint. When coalition partners publish the new information, the system first parses and maps it to the ontology. We use Apache Tika (http://tika.apache.org) as a parser that outputs XHTML. We developed a mapping between XHTML and our document ontology that automatically builds a particular document's ontology model based on the obtained document's XHTML. We store the resulting ontology models using Jena (http://incubator.apache.org/jena). Traditionally, OWL reasoners[12] operate by adding new facts to an ontology repository based on the relationships defined by the Resource Description Framework (RDF) and OWL (for example, type, subclass, property, or cardinality). To date, such reasoners primarily consider the model's structure and have limited facilities (such as exact text matching) to analyze node content. Such reasoners could infer that people with the last name "Smith" are members of the "Smith family," but would have no way of recognizing relationships for names that contained "smith," such as "Smithson" or "Jones-Smith."

We developed a new capability that dynamically constructs and reasons about type relationships based on document content and structure. We based this mechanism on the execution of a specific SPARQL[13] query over an OWL document model. It constructs the facts it infers from the query results and adds them to the ontology repository in the same way a traditional reasoner would. These facts let the system discover relationship types it couldn't find using traditional OWL reasoning methods.

As Figure 2 shows, when a user defines a new dynamic type relation, the relation context expression forms the `WHERE` clause of a SPARQL query (red text). For each instance or class in the model that matches the conditions of this `WHERE` clause, the mechanism will create a new set of facts (subject-predicate-object triples) according

```
CONSTRUCT
{
?subjInst
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<urn:relation#RegexMissionTitle>
}
WHERE
{
?subjInst
<http://ontology.ihmc.us/msoffice/msoffice.
  owl#hasSectionTitle>
?sectionTitle.
 FILTER regex(?sectionTitle, ".*mission.*", "i")
}
```

Figure 2. An example of the SPARQL construct. Here, the construct builds a new relation between documents related to a mission.

to the contents of the CONSTRUCT section (blue text). The relation context expression begins with the predefined variable ?subjInst, which refers to the information object being evaluated. The ? indicates a variable, and multiple references to a variable are bound to the same value. Notice that the CONSTRUCT section is populated with the subject ?subjInst (from the WHERE clause). In the example, the mechanism will search the OWL metadata for an information object for any node that has a hasSectionTitle property with a value matching the regular expression (regex) ".*mission.*". For each matching instance, say #doc1section1, the mechanism constructs a new triple, in this case,

```
#doc1section1 rdf:type
#RegexMissionTitle
```

The mechanism adds the results of this reasoning to the ontology metadata of the information Phoenix manages.

We've integrated this mechanism with the Phoenix IMS (see Figure 3). When a coalition partner publishes a full document, Phoenix can generate additional publications based on relationships to other types—for instance, it can publish the filtered information to certain clients registered only for a more specialized information type. The system can also extend the information itself with related available information from Phoenix's persistence information repository, which stores groups of correlated information, precomputed based on existing clients' subscriptions.

When a subscriber creates a new subscription, Phoenix can automatically identify additional subscriptions for related information based on a combination of the original subscription predicate and the relationships between the information types. The system can simply suggest the new subscriptions to the client or transparently fulfill them along with the original subscription. The system automatically joins the results for clients' queries with semantically related results. To process a semantic query, the system can either find the original result set and then discover related results based on precomputed relationships between information types; or find the result sets simultaneously and then correlate results based on their actual values.

Experiments and Results

We performed several experiments to evaluate the technologies we present here. Experiments with CBR focused on how well the text classifier correctly selected and classified relevant information for dissemination to clients. To evaluate the classifier's performance, we used data collected from the Internet Movie Database (IMDB; www.imdb.com) that involved movies rated PG, PG-13, and R. The collected data consisted of movie descriptions that were manually marked up with labels from six categories. We compiled the markup information into a list of sentences extracted from the movie descriptions and their corresponding category labels. We then used the individual sentences and their categories to train the text classifier, resulting in a case base with 1,946 total cases from the different categories: general violence (65), graphic violence (730), nudity (498), drug use (349), dark topics (298), and sexual content (6).

Next, we performed tests to measure the text classification's accuracy. These involved classifying sentences from the training set under two different conditions by varying the size of the case base the training set generated. Under condition A, we used all cases to test sentence classification, whereas under condition B, only 90 percent of the cases were available for the classification task; 10 percent were randomly selected and removed prior to testing. For condition B, we performed 10 tests to compute an average classification result. Table 1 shows the results under both conditions.

The classifier missed 16 out of 1,017 mark-ups under both conditions, meaning it assigned no label to the marked-up text from the training set, even though the sentence originally received a category label. It misclassified 310 markups under condition A, meaning the label assigned to the text was incorrect. Under condition B, misclassifications increased only slightly to 337. This shows that

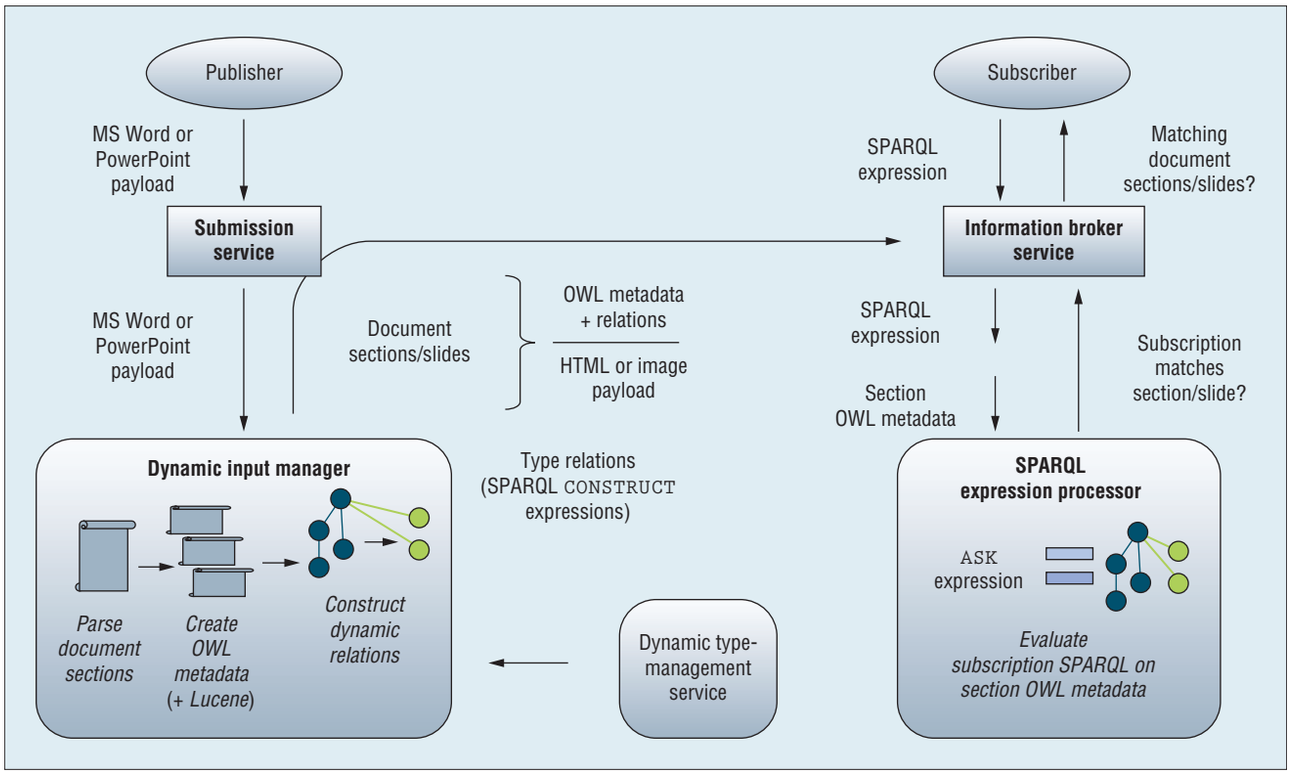

Figure 3. Integration of the dynamic information relation exploitation functionality with Phoenix. The diagram shows the processing steps for the published document on the path to the right subscribers.

some cases in the case base generalize well to multiple marked-up sentences in the training set, capturing both individual sentences and their meaning. The 16 missed classifications occurred in those markups that the system's text analyzing and case building component couldn't adequately translate into cases.

We also tested the performance of the semantic relationship mechanism integrated with Phoenix. Our implementation measured processing time at two key points: when the document is published and when it is delivered to the subscriber. Most of the heavy lifting occurs when a document is published. We specifically measured the duration of the new functionality in

- parsing the document to construct its OWL model,
- adding new facts to the OWL metadata from the dynamic relations, and
- publishing a new information object for each document section.

Table 1. Reclassifying marked-up text after the classifier was fully trained.

Test	Condition A	Condition B
Number of active cases in the case base	1,946	1,752
Number of documents processed	28	28
Total number of text markups in the training set	1,017	1,017
Total number of correct classifications	707	680
Total number of missed classifications	16	16

On the subscription processing side, we measured the time to compute the new dynamic relationship mechanisms:

- indexing the text in the OWL metadata and
- executing each subscriber's SPARQL `ASK` query over the publication's OWL model.

We conducted the tests and performance measurements on a single 64-bit Windows 7 Pro server, with a 2.27-GHz 8-core processor and 8 Gbytes of RAM. Both Phoenix and clients ran on the same host. Figure 4 presents the mechanism's performance using example Office document subscriptions.

The graph presents performance measurements while evaluating information to determine whether the document matches any subscription predicates, considering dynamically created relations. The greatest impact on performance is the number of unique subscription predicates (SPARQL expressions) that the system

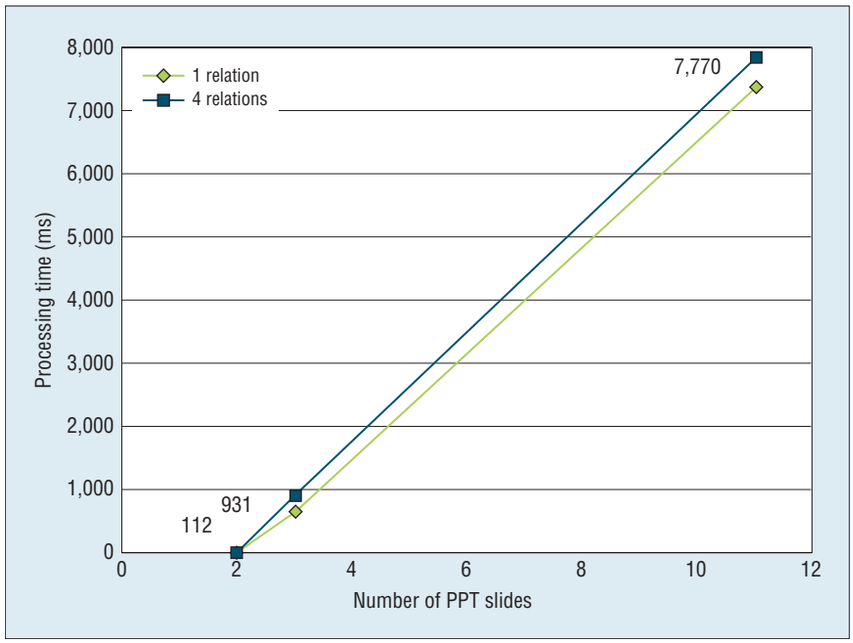

Figure 4. Processing time of subscription fulfillment. The processing time is proportional to the number of processed slides.

must evaluate on the information object's OWL metadata. The reason for this almost linear effect is the sequential evaluation of each SPARQL expression over the OWL information model.

Our extensions to Phoenix aim to automate information release to coalition partners in a controlled manner, based on the information's relevance to the requestors but preserving its sensitivity.

Performance demands for the new information management mechanisms must be controlled by policies[3] and addressed with more computing power—such as cloud computing. Policies can allow the system administrator and the users to set boundaries when the dynamic relation mechanism should be employed—for instance by limiting the number and the depth of relations exploited as dependent on the publisher's and subscriber's contexts. ■

THE AUTHORS

Andrzej Uszok is a research scientist at the Florida Institute for Human and Machine Cognition (IHMC). His research interests include ontologies, semantic reasoning, policy specification, agent systems, and transparent interoperability. Uszok has a PhD in computer science from the AGH University of Science and Technology, Krakow. He's a member of IEEE and ACM. Contact him at auszok@ihmc.us.

Larry Bunch is a research associate at the Florida Institute for Human and Machine Cognition. His research interests include facilitating human-agent teamwork through knowledge representation and reasoning as well as interactive visualizations. Bunch has a BS in computer science from the University of West Florida. Contact him at lbunch@ihmc.us.

Jeffrey M. Bradshaw is a senior research scientist at the Florida Institute for Human and Machine Cognition (IHMC), where he participates in the research group developing KAoS Policy and Domain Services, the Luna Software Agent Framework, and the Sol Cyber Framework. Bradshaw has a PhD in cognitive science from the University of Washington. He's a coeditor of the Human-Centered Computing department for *IEEE Intelligent Systems*. Contact him at jbradshaw@ihmc.us.

Thomas Reichherzer is an assistant professor in the Computer Science Department at the University of West Florida. His research interests include case-based reasoning, knowledge representation, and the Semantic Web. Reichherzer has a PhD in computer science from Indiana University. Contact him at treichherzer@uwf.edu.

James Hanna is a senior research engineer at the US Air Force Research Laboratory Information Directorate. His research interests include distributed information management, distributed policy enforcement, and advanced architectures to address the complex challenges of deploying information management in forward tactical environments. Hanna has a BS in computer science from the State University of New York (SUNY) Institute of Technology. Contact him at james.hanna@rl.af.mil.

Albert Frantz is a computer engineer and program manager at the US Air Force Research Laboratory Information Directorate. His interests include the Semantic Web, information management, planning, and command and control. Frantz has an MS in computer engineering from Syracuse University. Contact him at albert.frantz@rl.af.mil.

Acknowledgments

Portions of this work were completed at Enkia and funded by the US Air Force Research Lab (AFRL), Rome, New York, under SBIR Phase I and II grants, contract numbers FA8750-08-C-0110 and FA8750-07-C-0117. This work was also sponsored by another grant from the AFRL, contract number FA8750-10-1-0203. We acknowledge Ashwin Ram and Badri Lokanathan, formerly at Enkia, for their contributions to this body of work.

References

1. A. Uszok et al., "Rapid Creation and Deployment of Communities of Interest Using the CMap Ontology Editor and the KAoS Policy Services Framework," *Proc. 2nd Networked Digital Technologies Conf.*, Springer, 2010, pp. 451–446.
2. R. Grant et al., "Phoenix: SOA-Based Information Management Services," *Proc. 2009 SPIE Defense Transformation*

and Net-Centric Systems Conf., 2009; doi:10.1117/12.817911.
3. A. Uszok et al., "KAoS Policy Management for Semantic Web Services," *IEEE Intelligent Systems*, vol. 19, no. 4, 2004, pp. 32–41.
4. N. Suri et al., "A Dynamic and Policy-Controlled Approach to Federating Information Systems," *Proc. 2010 Military Comm. Conf.*, IEEE, 2010, pp. 2028–2033.
5. N. Ge, J. Hale, and E. Charniak, "A Statistical Approach to Anaphora Resolution," *Proc. 6th Workshop on Very Large Corpora*, 1998, pp. 161–171.
6. D. Gildea and D. Jurafsky, "Automatic Labeling of Semantic Roles," *Computational Linguistics*, vol. 28, no. 3, 2002, pp. 245–288.
7. S. Pradhan et al., "Support Vector Learning for Semantic Argument Classification," *Machine Learning*, vol. 60, nos. 1–3, 2005, pp. 11–39.
8. G.J.L. Kolodner, *Case-Based Reasoning*, Morgan Kaufmann, 1993.
9. D.B. Leake, *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, AAAI Press/MIT Press, 1996.
10. D.B. Leake and D.C. Wilson, "Remembering Why to Remember: Performance-Guided Case-Base Maintenance," *Proc. 5th European Workshop Case-Based Reasoning* (EWCBR 2K), LNAI 1898, Springer, 2000, pp. 161–172.
11. B. Smyth, P. Cunningham, and M. Keane, "Hierarchical Case-Based Reasoning: Integrating Case-Based and Decompositional Problem-Solving Techniques for Plant-Control Software Design," *IEEE Trans. Knowledge and Data Eng.*, vol. 13, no. 5, 2001, pp. 793–812.
12. D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*, Morgan Kaufman, 2008.
13. B. DuCharme, *Learning SPARQL*, O'Reilly Media, 2011.

cn *Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.*