The Briefing Associate: Easing Authors into the Semantic Web

Marcelo Tallis, Neil M. Goldman, and Robert M. Balzer, *Teknowledge*

The Semantic Web has clear benefits for information consumers, but for authors it typically means added work with no immediate payoff. This article describes a tool that eases the effort by automating markup within a popular COTS application.

The Semantic Web promises to expand Web services, both through automated software agents that perform what are now manual procedures and through new applications that are infeasible today. For this vision to materialize, Web documents must contain ontologically encoded information (or semantic markups) that software agents and tools can accurately and reliably interpret. Generating this encoding—at least in manually created documents—is a major challenge. Document markup is currently tedious and sometimes complex. Given that markup benefits accrue almost solely to agent-assisted content consumers, content producers have little motivation to undertake the extra effort.

While the research community has made considerable technical progress in supporting much of the Semantic Web life cycle, support for markup of manually composed documents remains quite primitive. The prevalent approach is to create specialized tools that support the association of semantic markups with preexisting document content[1,2] (see the "Related Work" sidebar). These tools provide a GUI that lets authors browse ontologies, find appropriate terms, generate syntactically correct markups, and associate them with portions of a document or, more often, with the document as a whole. This activity remains an extra effort that does not directly reward authors.

We are experimenting with a different approach. Rather than add semantic markups to completed documents, we have augmented a popular commercial off-the-shelf (COTS) authoring application with the Briefing Associate (BA), a tool that produces semantic markups as authors compose briefings. Our intent is to dramatically reduce the cost of producing semantically annotated documents and thus simplify the transition to the Semantic Web. We have also added analysis and synthesis tools that use these semantic markups during document composition to improve accuracy, quality, and production speed. Authors thus reap a direct benefit from creating documents with associated ontological encodings.

We've implemented a BA research prototype and tested it in a few simple domains. The most complex document composed with the BA so far produced semantic markup that describes more than 270 domain relationships.

Briefing Associate overview

The BA is a descendent of the Design Editor,[3] an application for producing visual, domain-specific design environments. Both tools are implemented as extensions of Microsoft's PowerPoint, which, as we describe later, is a key factor in our research.

How it works

The BA operates through the PowerPoint GUI. We extended the native GUI with a toolbar for adding graphics that represent a particular ontology's classes and properties. When authors compose briefings using these graphics, they indirectly construct DARPA Agent Markup Language descriptions of the briefing content. (For more on DAML, see www.daml.org.) The BA stores the semantic markup persistently, but not visually, within the PowerPoint document.

In addition to creating semantically grounded briefings, the BA exposes the briefing's semantic descriptions to external modules called analyzers, which will perform specialized services or analyses for the author. Such analyses might provide feedback to the author, extend or modify the briefing, or produce external documents derived from the descriptions.

1094-7167/02/$17.00 © 2002 IEEE

Related Work

There are currently several ongoing initiatives aimed at establishing a global semantic markup scheme for the Web. The oldest and most widely adopted is the Dublin Core Metadata Initiative (http://dublincore.org). The DCMI's goal is to facilitate electronic resource discovery on the Web. Its primary offering is the Dublin Core Metadata Element Set, a group of 15 elements—such as Title, Creator, Subject, and Date—that describe Web resources. The DCM Element Set is the de facto worldwide standard for describing information resources across disciplines and languages, and has already been translated into 25 languages.

Newer undertakings, such as the European Community's Ontobroker (http://ontobroker.semanticweb.org), its successor OntoWeb (www.ontoweb.org), and DARPA's DAML, go beyond DCMI goals. Rather than annotating electronic resources to merely facilitate their discovery, these projects aim to describe electronic and real-world entities using a machine-understandable language that lets autonomous software agents accurately understand and process their content.[1] We are developing the Briefing Associate (BA) under the DAML program.

There are two dimensions of requirements for semantic markup generator tools:

- *Description granularity* ranges from coarse descriptions, which relate a whole document with a set of predefined conceptual categories, to detailed descriptions of a document's content.
- *Description regularity* ranges from highly regular data, which is typically supported in relational databases, to descriptions of highly unstructured and irregular information, such as the content of newspaper articles.

We use these granularity and regularity dimensions to compare the BA with other tools for generating semantic markups. Our aim with the BA is to offer detailed descriptions of irregular and unstructured documents.

- The Nordic DC metadata creator (www.lub.lu.se/cgi-bin/nmdc.pl) is a metadata editor for the DCMI. It consists of a Java applet that displays a form for users to input DCM Element Set values. The Nordic DC then generates a syntactically correct encoding of these values that a user can attach to the described document. This tool corresponds to less elaborated forms of semantic markups, such as coarse descriptions based on a predefined set of conceptual categories.
- Klarity (www.klarity.com.au) is another type of metadata generator that supports the DCM Element Set. Klarity automatically generates metadata for HTML pages based on concepts it finds in the text. It uses statistic methods to allocate values based on concepts it identifies in the "seed" or exemplar documents related to the target concept. Like Nordic DC, Klarity generates coarse metadata descriptions of documents.
- ITtalks[2] is a portal for announcements about IT-related talks, seminars, and colloquia, developed under the DAML program. Although not its main focus, ITtalks generates DAML descriptions from the talks contained in its database. In this sense, ITtalks generates descriptions from highly structured data.

Two tools closer to the BA's scope are the KA[2] initiative's Annotation Tool,[3] developed under the Ontobroker project, and the Shoe project's Knowledge Annotator.[4] Both tools offer a GUI for authoring and attaching semantic annotations to Web documents. They also offer context-sensitive instances and ontology browsers that facilitate the creation of semantic descriptions. A second incarnation of KA[2] can semiautomatically generate annotations using a lexical text analysis and a vast word and domain lexicon

Unlike these approaches, the BA generates markups as a byproduct of constructing the document. Hence, it does not require extra work from authors. Also, because the BA's semantic annotations are embedded in the original document—rather than inserted in a second step using a different tool—modifying the original document does not erase the existing annotations. On the other hand, the BA approach might prove inadequate for marking up

- existing documents that do not use the BA conventions for representing ontological relationships, and
- documents whose types are not supported by the underlying COTS product.

Although the KA[2] annotation tool's semiautomatic markup generation simplifies the production of semantic annotations, it still constitutes an extra activity because users must check and revise the generated annotations. The KA[2] approach is also limited to textual documents that contain enough information to infer their semantic relationships. This limitation might exclude briefing documents because they usually contain diagrams that are not explained within the text.

References

1. T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, May 2001, pp. 34–43; www.sciam.com/2001/0501issue/0501berners-lee.html (current Dec. 2001).
2. R. Scott et al., "ITTALKS: A Case Study in the Semantic Web and DAML," *Proc. First Semantic Web Working Symposium,* 2001; www.semanticweb.org/SWWS/program/full/paper41.pdf (current Dec. 2001).
3. M. Erdmann et al., "From Manual to Semi-Automatic Semantic Annotation: About Ontology-Based Text Annotation Tools," to be published in *Linköping Electronic Articles in Computer and Information Science*, vol. 6; available at www.ida.liu.se/ext/epa/cis/2001/002/tcover.html (current Dec. 2001).
4. J. Heflin, J. Hendler, and S. Luke, "SHOE: A Prototype Language for the Semantic Web," to be published in *Linköping Electronic Articles in Computer and Information Science*, vol. 6; available at www.ida.liu.se/ext/epa/cis/2001/003/tcover.html (current Dec. 2001).

Implementation

We regard our choice to implement the BA as an extension of the PowerPoint platform not as an implementation detail, but as central to our research for two key reasons:

- PowerPoint gives us a higher-level platform for building a briefing tool than generic middleware such as COM/Corba and GUI widget libraries. It offers an extensive graphical base for representing a briefing's visual content and support for retaining nongraphical data, such as DAML markup persistently within its documents. PowerPoint also offers an extensive WYSIWYG user interface for view-

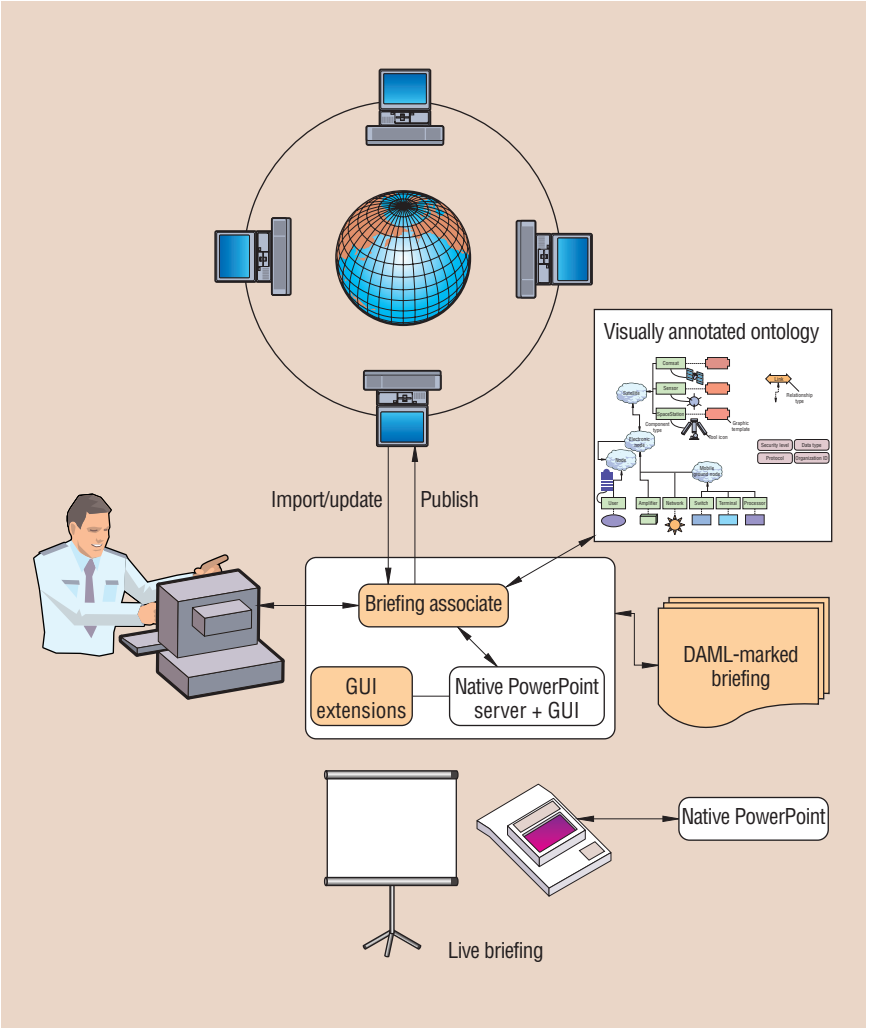


Figure 1. The BA's software architecture. An author selects a visually annotated ontology and creates a briefing using the BA's augmented PowerPoint user interface. The resulting document can be presented as a slideshow with native PowerPoint software. The document retains semantic markup tied to its visual elements as a byproduct of the author's actions.

ing and editing a briefing's visual content. To accommodate DAML-aware briefings, the interface requires only an extension, not a redesign or reimplementation.

- PowerPoint is the most widely used product for authoring briefings. An author can thus adopt the BA without learning a new production environment and without sacrificing familiar features. Briefing authors familiar with PowerPoint can continue to use the native GUI tools, menus, and direct-manipulation actions to edit visual content. The BA simply extends the interpretation of these tools and actions and adds tools tailored to a target ontology.

We programmed the BA primarily in Visual Basic. For PowerPoint 2000/XP, the BA is a COM add-in that receives "events" as the author creates, opens, closes, and modifies briefings. As a client of PowerPoint, the BA can also navigate through a briefing, paint analysis feedback directly onto it, and associate DAML descriptions with graphical items. For efficiency reasons, the BA runs entirely as an in-process component—it is incorporated into the PowerPoint process itself. With both the BA and PowerPoint residing in a single operating system process, communications are extremely efficient. Although we could achieve greater efficiency by implementing the BA in C++, the Visual Basic code's performance has been acceptable to date.

The BA components

Figure 1 shows the BA's architecture and major information flows. The BA appears to authors as an ordinary PowerPoint presentation editing environment, with a few extra tool buttons that create graphics representing instances of classes and properties from a target ontology. Portions of the briefing created with these tools are automatically annotated with nonvisible DAML markup. Authors can augment the briefing by importing preexisting DAML markup, which is automatically rendered according to the graphic conventions of the target ontology. Authors lose none of PowerPoint's native capabilities. The DAML markup persists with the briefing document as part of the saved presentation or slideshow file. Authors can also publish the document's markup independently of the graphic content using the standard XML encoding for DAML.

The semantically grounded briefing (SGB) editor appears to authors as a modestly extended user interface to PowerPoint. The extensions are tools and menu items for inserting graphics that represent class and property instances in a target ontology. The BA itself is not tied to any specific ontology; it creates extensions based on a visually annotated ontology (VAO).

The VAO editor provides a graphical view of an ontology. A domain graphic designer adds graphic annotations to this ontology, specifying a standard rendering for its classes and properties. A single VAO will typically be reused by multiple authors to create many briefings. An author selects a VAO for a briefing from a library of registered VAOs. The BA records the choice with the briefing's markup. Whenever the BA is focused on a briefing, the PowerPoint GUI is augmented according to the chosen VAO's dictates.

The semantic-content import and update component lets authors create new graphic briefing content from preexisting DAML descriptions. The VAO assigned to a briefing determines an initial rendering for the imported content. Authors then use the native PowerPoint GUI to manually edit the layout and other graphic details used in the rendering.

As with a VAO, a domain expert creates an analyzer for reuse across multiple briefings by multiple authors. An analyzer provides a service based on the DAML description associated with a briefing. Such a service might provide feedback on a briefing's completeness or consistency, or its adherence to established guidelines.

SGB editor

The SGB editor lets authors create original content, and edit both original and imported content. Authors control the SGB editor through a combination of standard PowerPoint interface actions, additional GUI tools and menu items, and direct manipulation. The native PowerPoint GUI is completely functional, and all user preference settings are preserved.

The editor is guided by a VAO, which associates a graphic template with each class and object property in a single ontology. A class template might be either an image or a PowerPoint shape. A property template must be a PowerPoint connector. Because a connector can be attached to (at most) two other graphic objects, it is a natural rendering for a property. Disconnected property graphics are interpreted as properties for which the domain or range instances are not (yet) specified.

The editor adds a toolbar to PowerPoint's GUI that has a tool for each of these classes and properties. By selecting a tool, authors can insert a copy of the graphic template anywhere in the briefing, as if they selected a native PowerPoint tool for adding a shape or connector. These ontology-aware tools simultaneously associate the ontology class or property with the new graphic.

Figure 2 shows a screenshot of the editor in a satellite communications domain. Everything in the figure is part of the GUI, with the exception of the callouts highlighting specific elements. The screen's center contains a slide depicting a satellite communications configuration. The various labeled shapes represent class instances from the ontology: satellites, terminals, switches, processors, and users. They are connected by arrows representing communication links, which are this simple ontology's sole object property.

The ontology toolbar appears in the second toolbar row (upper right). To its left is a box with a drop-down list that displays the current ontology's name (in this case, "Satellite Com"). To start a new briefing, the author chooses an ontology from this list, which triggers the creation and display of the appropriate ontology toolbar. To manipulate graphical aspects of the briefing's class and property instances—such as positioning, resizing, and attaching or detaching connectors—the author uses PowerPoint's native mouse gestures or keyboard shortcuts.

In addition to graphic templates, a VAO lists any analyses available to the author. The presence of analyses in the VAO is reflected by additional menus in the PowerPoint menu bar. In Figure 2, the author requested a topology analysis from the "Designer studies" group. The result of the analysis is a list of reports, displayed in a separate window in the upper left. Each report consists of a textual explanation and optional graphical feedback. In this example, there is just one report. Its explanation reads, "User U3 is directly connected to User U2." When the author selects a report, its graphical feedback is displayed. In this case, the feedback highlights the communication link between U2 and U3 as a thin red arrow. When the author deselects the report or closes the window, the highlighting is reversed and the link returns to its normal state.

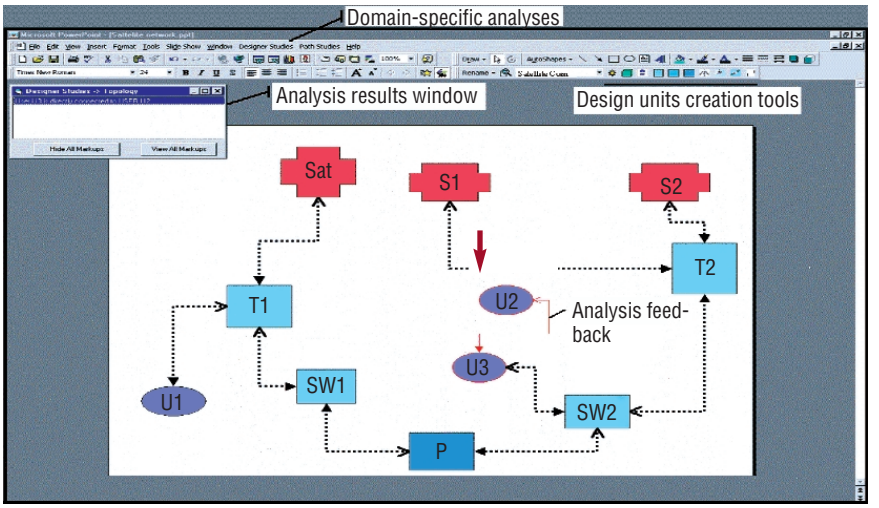


Figure 2. The SGB editor's GUI in the satellite communications domain.

The editor lets authors edit datatype (literal) properties of class instances through a dialog box interface. In a briefing, every graphic associated with an ontology class has an Edit Attribute Values menu item in its context menu. The author selects this item to display a *tabbed dialog* for the selected instance, with a tab for each datatype property applicable to that instance. Each tab provides a text, radio button, or checkbox interface (depending on the property's range type and cardinality) that lets authors view and set the property's value. Figure 3 shows the dialog for a sensor satellite. The dialog contains a tab for each of 10 datatype properties in the ontology that have "sensor satellite" or one of its superclasses as a domain. The editor uses analogous dialogs to gather the parameter values for parameterized analyses.

The editor also extends PowerPoint's cut, copy, and paste operations. Whenever authors

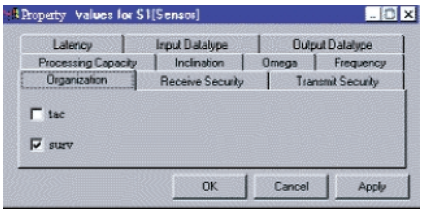


Figure 3. The dialog for a sensor satellite. Dialogs are displayed on demand and contain tabs for each datatype property applicable to the selected object.

cut or copy a graphic to the clipboard, its markup accompanies it. When they paste that graphic into its original briefing or into another briefing using the same ontology, the markup is restored. There are actually two pasting modes. One ("paste as new instance") creates a new markup object instance, copying the class and datatype property values from the clipboard. The other ("paste as proxy") simply adds a graphic to the briefing and treats it as an additional instance reference already referred to elsewhere in the briefing. Use of such proxies lets authors distribute information about a single object across multiple slides.

VAO editor

The VAO editor lets a domain graphic designer add graphic annotations to any DAML ontology. Like the SGB editor, the VAO editor is implemented as an extension of PowerPoint—in fact, technically, VAO is simply the name we give the SGB editor when it is being driven by a VAO for ontologies. Using the VAO editor, a graphic

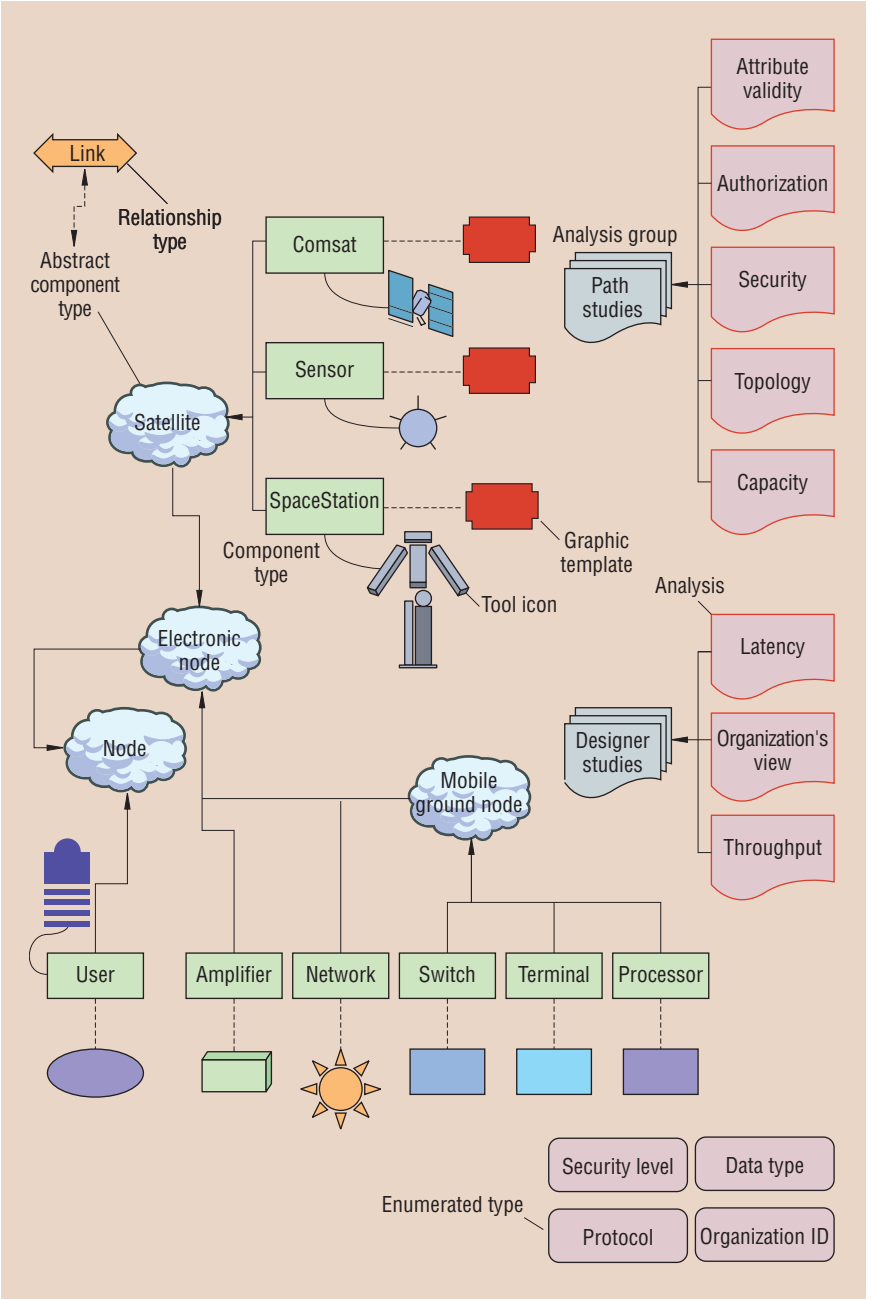


Figure 4. Visually annotated ontology for the satellite communication domain. A VAO associates graphic templates with ontology classes (green rectangles) and properties (orange double arrow). It also specifies the supported analyses (right side).

designer selects classes and properties from an ontology and associates them with graphic templates. When the designer imports ontology *X* into the VAO editor, it lays out *X*'s classes and object properties, graphically depicting their hierarchical relationships (their subclass and subproperty properties). The designer can then create an arbitrary PowerPoint graphic for some or all of them. The VAO created by this activity establishes a rendering convention for briefings.

The designer also has the option of assigning toolbar icons to the graphically annotated classes and properties. When a VAO contains a graphic template, but no toolbar icon for a class or property, the SGB editor simply uses a scaled-down replica of the graphic template as the toolbar icon.

Figure 4 shows the satellite communications VAO used in Figure 2. The green rectangles labeled "Comsat," "Sensor," "User," and so on depict the leaf ontology classes. The shapes attached to them by dashed connections are graphic templates assigned by the designer. In Figure 2, copies of these graphic templates depict instances of the associated classes. Designers can use any of PowerPoint's native autoshapes as a graphic template and format them however they want. They can also use an image as a graphic template. The light-blue clouds ("Satellite," "Electronic node," and so on) represent non-leaf classes. The orange arrow ("Link") defines the ontology's sole object property, and the attached dashed, double-headed arrow is the graphic template for the "Link" property

To create tool icons for the instantiation toolbar, authors connect classes and properties to graphics using a curved-solid connector. Like graphic templates, authors can select any PowerPoint graphic as a tool icon.

In addition to providing visual annotation, a designer can use the VAO editor to specify initial values for the datatype properties applicable to any class. The interface is a tabbed property-editing dialog, identical to those the SGB editor uses (see Figure 3). When a VAO specifies an initial value for a property of a class, the SGB editor assigns that value to instances of the class as they are created from the ontology toolbar.

Figure 4 shows the specification of two analysis groups, "Designer studies" and "Path studies," and the eight analyses they contain. The color and styling of an analysis' border specifies how the BA will render graphical feedback supplied with analysis reports. For example, Figure 2's U2 to U3 connection report contained feedback specifying that the link between them be highlighted. The highlighting was rendered as a thin red line because the "Topology" analysis border is a thin red line. Analogously, an analysis label's text characteristics—font, face, size, and color—specify the textual characteristics of any feedback text an analysis asks to paint on a briefing.

Although an analysis can specify graphical feedback characteristics in great detail, deferring to the VAO's "convention" simplifies the analysis programmer's job. It also leaves the choice of graphical characteristics to graphic designers, who can choose schemes that avoid conflicts between graphic feedback conven-

tions and the VAO's rendering conventions.

Although we have focused on the VAO editor adding annotation to a preexisting ontology, it is in fact a full-fledged ontology editor and can be used to declare new classes and properties.

Content import and update

To date, we have created a series of ontology-specific import and update components. In the following, we describe our planned generic component.

Our content import component will let an author contact and query DAML-aware agents, including search agents. It will then accept the query results as DAML descriptions and incorporate them into the presentation, retaining the imported descriptions and the agent and query that produced them.

The component will graphically render these descriptions as specified in the applicable VAO, thereby creating the imported content within the briefing. Authors must then adjust the graphics' sizes and positions using the native PowerPoint GUI to produce an acceptable layout.

The component will also let authors update information on demand. Using the retained queries, the component will requery source agents to retrieve updated content and render an updated version. The component will visually correlate the two versions on demand. Authors can then incorporate the updated version as a whole or selectively incorporate only changed information.

Analyzers

Analyzers are external, executable modules that process the briefing content's DAML descriptions to provide an analysis, a synthesis, or some other service. An analyzer can be implemented to execute from within the PowerPoint process, as a separate process on the same host, or on a different host (through DCOM).

Analyzer interactions. Each analysis is associated with a particular ontology. A VAO factors an ontology's analyses into groups. The grouping is reflected in menus that the SGB editor adds to PowerPoint's menu bar. When an author requests an analysis, the BA connects to, or launches, the module implementing that analysis and passes it a reference to the target briefing, along with any author-provided analysis parameters. The analyzer subsequently sends the BA a set of reports that constitute the results of the analysis. The BA presents the reports to the author.

For a *snapshot* analysis, the analyzer's responsibility ends with report transmission. With an *incremental* analysis, however, the analyzer updates its reports as the author modifies the briefing. The updates end when the author either closes the analysis report viewer or closes the briefing. To support incremental analyses, the briefing reference that the BA sends the analyzer provides direct access not only to the briefing's content, updated as the author edits, but also to events broadcast by the SGB editor as changes occur.

An analyzer can also use a briefing reference to gain direct access to PowerPoint's detailed graphic model of a briefing. However, analyzers are typically interested only in the DAML descriptions that encode the briefing's semantic content.

> The BA constitutes a new paradigm for generating semantic descriptions that minimizes the overhead typically incurred when authors annotate documents with semantic markups.

The DAML content conveyed to an analyzer could be encoded in an ontology-independent form, such as XML text or an RDF object model (www.w3c.org/RDF). However, reliance on such an encoding would make analysis programming tedious and would generally result in inefficient analyzers as well. We have chosen instead to provide an analyzer with an ontology-specific interface to briefings. For each ontology, the BA automatically generates a COM *type library*. This type library reflects a straightforward mapping between ontology classes and properties and COM's corresponding modeling concepts (classes, interfaces, and properties). Most widely used software development environments for Windows provide a declarative way to import such type libraries, automatically building the client-side code needed to interact with servers of the library-defined objects. Analyses can then be written in a context in which the ontology's classes and properties appear as first-class types and methods of the development language.

Analyzer examples. One analyzer we have developed exports a briefing's markup in the XML encoding of the DAML+OIL language (www.daml.org/2001/03/daml+oil.daml). Using a generic briefing ontology, it also exports meta information (author, date, size, and so on) and all titles and text that appear in the briefing.

One of the analyzers for the satellite communications ontology computes communication latency across paths between a pair of components selected by the author. For each possible path, a report is returned containing its latency. The graphic feedback accompanying a report highlights the components and links on that report's path.

Author benefits

Although implementing the BA imposes no extra impediments or costs on producing PowerPoint briefings in the standard way, it's unrealistic to expect authors to use the BA's extensions based solely on the prospect of future benefit to others. The BA includes several enhancements that offer authors immediate benefits:

- *Graphic templates*. The BA simplifies briefing construction by offering readily available graphic templates that authors can repeatedly use to represent domain objects.
- *Analysis, synthesis, and other services*. The BA's analyses exploit the briefing content's semantics to provide services to authors while they're editing their briefings. Although such analyzers are not necessarily tied to the Semantic Web, the availability of DAML encoding of briefing descriptions means that Web-based agents can be used as part of the implementation of these services.
- *Potential time savings*. The BA's extensions for importing and visualizing preexisting DAML descriptions can save authors significant time in constructing certain kinds of briefings. Because we've designed these facilities to rely on Semantic Web queries, they leverage, rather than bootstrap, the Semantic Web vision.
- *Automated updates of imported content*. The BA automatically updates content that originates in the Semantic Web. As with import and visualization, this leverages the Semantic Web vision.

These BA author enhancements embed the Semantic Web life cycle into the briefing creation process so that authors can also enjoy two benefits of semantic markup.

First, ontology-based annotations will turn briefings into reusable resources. Authors can publish new content, as well as novel aggregations of imported content, in a form accessible to DAML-enabled agents. Linking the briefing's graphic content to the semantic content fosters reuse of both visual and semantic material. Although this does not directly benefit the publishing author, it does benefit any author who imports published material.

Second, the BA's automated content update facilities will transform briefings from information snapshots—which decline in value as the information becomes dated and obsolete—into renewable information resources that are automatically updated.

The BA constitutes a new paradigm for generating semantic descriptions that minimizes the overhead typically incurred when authors annotate documents with semantic markups. We have several additions to it planned or in progress.

We are currently augmenting the repertoire of visual metaphors for representing ontological relationships. Currently, the SGB editor only accepts and displays datatype (literal) properties through textual dialog boxes. We are extending the BA to allow the representaton of some properties as visual attributes of their corresponding graphic template. Authors can then edit those property values by changing the corresponding graphic attributes. For example, a graphic's label can represent a text property (such as a name), the graphic's size can represent a numeric property (such as the length of a queue), and the graphic's color can represent an enumerated datatype (such as a state).

We also plan to extend the BA's capabilities in representing relations between instances. The current BA can only represent a relation between two instances using an arrow to connect their graphic representations. This mechanism constrains authors to specifying relations between a pair of instances that are both included in the briefing's visual content. We plan to implement an alternative mechanism for specifying relations with instances external to the briefing. We also plan to add a tool to search Semantic Web content for references to preexsisting instances.

We are also applying our paradigm to Microsoft Word. In this case, instead of relying on a visually annotated ontology, we'll use a textually annotated ontology that might include text fragments for representing ontological relationships.

One outstanding question concerns the BA's applicability in a production environment. In our tests, we have demonstrated that we can use the BA to compose documents that fulfill our requirements in terms of its semantic descriptive power. However, these documents were composed to test the BA rather than to create briefings for a real audience. We have yet to demonstrate that it is practical to compose useful briefings that also convey useful semantic information. ■

References

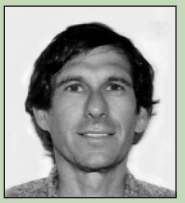
1. M. Erdmann et al., "From Manual to Semiautomatic Semantic Annotation: About Ontology-Based Text Annotation Tools," to be published in *Linköping Electronic Articles in Computer and Information Science*, vol. 6; available at www.ida.liu.se/ext/epa/cis/2001/002/tcover.html (current Dec. 2001).

2. J. Heflin, J. Hendler, and S. Luke, "SHOE: A Prototype Language for the Semantic Web," to be published in *Linköping Electronic Articles in Computer and Information Science*, vol. 6; available at www.ida.liu.se/ext/epa/cis/2001/003/tcover.html (current Dec. 2001).

3. N. Goldman and R. Balzer, "The ISI Visual Design Editor Generator," *Proc. 1999 IEEE Symp. Visual Languages*, IEEE CS Press, Los Alamitos, Calif., 1999, pp. 20–27.

The Authors

Marcelo Tallis is a research scientist at Teknowledge Corporation. His research includes the development of tools that support creation of semantic markups for the Semantic Web and support COTS product wrapping to extend their functionality and provide safe and secure execution environments. He previously worked at the University of Southern California's Information Sciences Institute researching knowledge acquisition for knowledge-based systems. Tallis received a PhD in computer science from the University of Southern California. Contact him at Teknowledge Corp., 4640 Admiralty Way, Ste. 231, Marina del Rey, CA 90292; mtallis@teknowledge.com.



Neil M. Goldman is a senior research scientist at Teknowledge Corporation, where he develops middleware to support mediation of communications between COTS software components, with a focus on enhanced host-based security. He is also working on the integration of COTS desktop tools with the semantic Web. He previously worked at the University of Southern California's Information Sciences Institute researching application of artificial intelligence to software engineering and formal software specification languages. He received a BS in mathematics and an MS and PhD in computer science, all from Stanford University. Contact him at Teknowledge Corp., 4640 Admiralty Way, Ste. 231, Marina del Rey, CA 90292; ngoldman@teknowledge.com.

Robert M. Balzer is a senior research scientist and the chief technical officer of Teknowledge Corporation. He also directs Teknowledge's Distributed Systems Unit, which combines artificial intelligence, database, and software engineering techniques to automate the software development process. Current research includes wrapping COTS products to integrate them, provide safe and secure execution environments, and extend their functionality. He is also researching instrumenting software architectures and generating systems from domain-specific specifications. He previously worked for the Rand Corporation, and later helped establish the University of Southern California's Information Sciences Institute, where he was director of the Software Sciences Division. He was also a professor of computer science at USC from 1972 to 2000. Balzer received his BS, MS, and PhD in electrical engineering from the Carnegie Institute of Technology in Pittsburgh. Contact him at Teknowledge Corp., 4640 Admiralty Way, Ste. 231, Marina del Rey, CA 90292; bbalzer@teknowledge.com.