In January, we talked about search and structure – tools to find and reveal the structure in seemingly unstructured, passive data – maps of a concept space, hierarchies of people or things. Now we turn to the more complex world of entities that have precise and interdependent relationships. You can’t just map those relationships; you have to model them and define how changes ripple through the model when one item in it changes.

The world of search and structure is static, but the real world we want to model is active. Actions and events have consequences; a change here drives more changes over there. The butterfly effect – unpredictable as it seems – results from changes rippling through a system. What appears to be randomness is more often the result of fuzzy measurement: Build a good enough model (or simulation) and you should be able to predict the impact of the changes from initial conditions.

“We’re trying to close the gap between the reality and the model,” says Microsoft alum Charles Simonyi of Intentional Software (even though he dislikes the word “model;” see page 12).

Any software application expresses some implicit model of the world, but often that model is no more visible than hidden meanings in the world itself; i.e. the software doesn’t explain anything. Explicit models in IT potentially enable three things:

- We can look at an IT system to see the impact changes may have, whether on the results of a particular action or on the functioning of the IT system overall.
- We can map from one model to another.
If we have a proper correspondence between them, we can make changes in a model and see them automatically reflected in software that implements the model.

In the issue that follows, we explore the use of software models: that is, both models of software, and models in software. These tools will variously help users be more efficient in creating and maintaining their code, and more effective in modeling the real world so that they can either automate processes to do what they want, or understand processes that are already happening. Anyone building any kind of software these days needs to build in the capability to communicate not just physically but semantically with the rest of the world, without necessarily knowing what is going to be in that world. (That is, you have to describe yourself well enough for strangers to understand you.) While the code tools are of interest to software developers, the broader modeling tools intrigue everyone from Coca-Cola trying to understand its own reseller network, to various US government agencies trying to model the behavior of terrorists and to detect instances of it amidst mountains of innocuous data.

From maps to models: From categories to active relationships
Oddly enough, it’s actually simpler to represent software code, which you would think would be complicated, than to represent the data described by the code. That’s because code is concrete; that’s all there is. By contrast, data may have lots of unrepresented meaning – some of which is represented in applications, to be sure, but may not be explicit. A function either calls another function or doesn’t. Deciding whether it should call that other function…that’s part of the data/meaning (or semantics) problem. Finally, there’s the challenge of modeling different views of the same data – perhaps not an issue in principle, but a real requirement in practice. (In theory, there’s no difference between theory and practice, but in practice, there is.)

We discuss these issues and the companies offering solutions in the order of increasing miraculousness (see the illustration on the following page), from tools that model software dependencies (modeling code), to tools that help IT people reconcile multiple schemas.
referring to the same data (modeling data semantics), to tools that create or execute models of all the complexity of the real world (modeling reality). The underlying technology ranges from databases that hold information about code, to code that holds information about databases, to inference engines that can propagate changes through a model or drive the creation of code. Along the way, we look at practical uses for this kind of approach.

**Modeling IT Systems: The Structure of Code**

Ironically, one of the most urgent things for IT users to model is our own systems. (If the proper study of mankind is man, the proper study of software is software.)

Using models in IT is not a new idea—or a new endeavor. With a little bit of discipline and documentation, say many cios, the tools we describe below would not be necessary. (But that’s unfair: It’s not a lack of discipline that makes one company’s data model differ from another’s.) Others point to all the automatic programming tools and projects that litter the IT landscape, and wonder why the new generation is any different. There are three differences from the world in which those tools were developed:

- The tools are getting better, with XML and other standards making data and code more “visible” and self-explaining, and tools such as Semtalk from Semtation in Germany adding actionability to graphical tools such as Microsoft Visio. And of course there are all the tools described in this issue (if that’s not too recursive a thought!).
- The machines are getting better, with capacity continuing to double every year or so.
- The volume, complexity and heterogeneity of IT systems is growing rapidly, making better models more necessary than ever. (No one now uses the word “silo” in a positive way.)

In a modern company, cfos as well as cios care about IT, while business-unit employees use IT and want to share data with other units. A business-unit manager cares about uptime and may even want to model the impact a change in her unit’s demands might have on the IT infrastructure as a whole...if only to point out that the impact would be minimal.
Representing IT systems

Actionable, comprehensive models of IT systems are hard to find. Ask any IT manager what’s running, where it’s running, how the pieces depend on each other, and, most importantly, what it’s doing, and he’ll tell you he knows. But poke around a little and you’re likely to find an undocumented mess. Or you’ll find things documented locally, but how one system depends on another is unclear. This information is useful for a variety of reasons, ranging from preventing and fixing problems that affect critical business services, to managing upgrades or knowing what you are paying license fees for, to figuring out how to re-engineer your company’s processes to reflect new business needs.

You’d think that IT systems would be one of the easiest things for software to know about, but in fact few programs “represent” themselves effectively. (That’s what UDDI is all about. However, UDDI is not a model, but just a directory with listings and some pointers.) Most programs are “about” what they contain or do, and do not focus on describing themselves. They come as a set of executable files, and perhaps some “read-me” or installation files, but they don’t put themselves in context – either in a taxonomy (“I’m a CRM application”) or in terms of their relationships with other systems (“I read SQL files off that database on that server and produce data to drive the payroll application”). Increasingly, applications do contain some such descriptions, but they are mostly annotations, not in machine-interpretable form and not “actionable.” That is, there’s nothing that another program could act on.

Whom do you talk to? and in what language? are key questions in today’s distributed application environments and will be more so in the coming days of Web services, when applications are expected to communicate with “foreign” applications.

Expecting an easy answer is like expecting to know what you’re eating via a description of the major brands of food items. Most software, like most food, is actually
home cooking based on branded products. It may contain some brand-name ingredients, but much of it has been stewed, chopped or diced, spice-added, and baked into firmness. Figuring out the ingredients so as to improve the recipe is the challenge facing every IT manager – and especially every new IT manager trying to figure out what his predecessor created. Companies who overbought during the boom and companies who have legacy systems built over the years are now trying to rationalize their operations, but first they have to figure out what they’ve got.

Change is a constant in the IT world (reflecting the real world), but it’s hard to change things you can’t understand and manipulate. The companies below all model IT systems to produce actionable information. They reflect the trade-offs among automated discovery, breadth of coverage (what environments and applications they “discover”), and the depth of what they analyze (the granularity of dependencies). Troux is not as highly automated but it has the greatest breadth of coverage and granularity. Collation is very automated and granular, but covers only a limited set of environments. And Relicore is perhaps the most powerful in general and the strongest on metrics, and it operates in real time.

---

### A GRID (NOT A MODEL)

<table>
<thead>
<tr>
<th>Company</th>
<th>Model about</th>
<th>Generated how?</th>
<th>Descriptive or active</th>
<th>Level of granularity</th>
<th>Breadth of coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Modeling IT systems</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collation</td>
<td>J2EE installations</td>
<td>Auto (polling)</td>
<td>Descriptive</td>
<td>Configurations, dependencies</td>
<td>From apps to infrastructure</td>
</tr>
<tr>
<td>Troux</td>
<td>IT installations</td>
<td>Manual and auto</td>
<td>Descriptive</td>
<td>Applications, dependencies</td>
<td>Arbitrarily broad</td>
</tr>
<tr>
<td>Relicore</td>
<td>Operating IT systems</td>
<td>Auto (agents)</td>
<td>Descriptive, real time</td>
<td>Applications, deps, metrics</td>
<td>Most software</td>
</tr>
<tr>
<td><strong>Modeling data and schemas</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contivo</td>
<td>Data</td>
<td>Manual</td>
<td>Descriptive</td>
<td>Down to data elements</td>
<td>Data-centric</td>
</tr>
<tr>
<td>SchemaLogic</td>
<td>Schemas</td>
<td>Manual</td>
<td>Descriptive/ workflow</td>
<td>Down to data elements</td>
<td>Data, schemas</td>
</tr>
<tr>
<td>Unicorn</td>
<td>Data &amp; schemas</td>
<td>Auto/manual</td>
<td>Descriptive/ active</td>
<td>Down to data elements</td>
<td>Data, schemas, business rules</td>
</tr>
</tbody>
</table>

No magic quadrants! This table is intended more as an aide-memoire than as a standalone model of the space, which is way too heterogeneous to represent in only two dimensions. Some of the tools covered fit awkwardly within the feature set outlined, but we have attempted to provide flavor rather than rigor.

Definitions:

**Auto vs. manual:** Can the tool “discover” the environment by looking at it? Or do humans have to tell it what’s there? (It can “look at” it from a central point – polling – or by installing agents locally.)

**Descriptive vs. active:** Does the tool do anything, or does it simply provide information for a human to act on?
Troux: Troux-bloux Texas practicality
Troux was founded in April 2001 by Hank Weghorst, who had previously founded Ventix (knowledge management), Question Technologies (B2B online selling tools) and Media Logic (graphics). That’s a logical background for Troux, which provides what it calls software “blueprints,” visualizations (and reports) that help developers, IT managers and other business managers understand not just what they’ve got but also what depends on what. Troux is based in Austin, with funding from Austin Ventures, which also funded Ventix and Question.

Troux’s goal is to go beyond typical software inventory packages, which simply list what you have (and perhaps how much it’s used) in order to provide more complex information on all kinds of software, both vendor packages and in-house applications and enhancements. This information is helpful in software development, reuse of software modules, etc. “You need to know what the software can do and whom it can talk to before you can figure out how to extend or replace it,” says Weghorst. “What information does it produce? What information does it require?”

While software inventory tools are targeted at people managing IT systems, Troux is designed for people developing or changing them, or even business people trying to understand a company’s IT installations and their impact on the business. Software inventory reports tend to be primarily financial and statistical: Which products are used how many hours by which departments. Troux’s are more qualitative and structural: This product uses these kinds of data from that application, and sends out this other kind of data. Or, this application produces reports that are not used by anyone, anywhere. Or, this same function is performed by these three servers which are therefore candidates for consolidation.

While Troux is not as automated in terms of discovery of the “inventory” as, say, Collation (below) or some inventory packages, its strength lies in its ability to collect and aggregate all kinds of information from all kinds of sources (including users), and then to feed it back in a way that’s meaningful to all kinds of people in all kinds of functions.

Culture follows code
One of Troux’s leading installations is at Northern Trust, the Chicago-based money-management firm, which started looking at the area based on several efforts to bul-
letproof its own systems. Vp technology Michael Shapkarov says, “We wanted to have a greater understanding of our underlying components and their dependencies.” After checking out the field, the technology group settled on Troux.

The big benefit, says Shapkarov, is that IT can now make changes intelligently; it understands the dependencies between systems and can minimize negative impacts – or spread the benefits of an enhancement through to every group that could make use of it. The system is not magic, he notes; it can discover the actual dependencies of, say, an application on a particular database, but it can’t automatically discover that the hand-coded client application in Department A is the same as the one in Department B. But once the user has filled in the blanks that the software cannot, it offers great value by managing a broad set of data from across an enterprise. Alice may not be responsible for a problem in Juan’s department, but it may still affect her department until it’s solved. “Troux collects data from disparate sources,” says Shapkarov, “and then keeps it unified. Before we had various technology teams with their own source of information, sometimes with an overlap.”

Troux’s primary output comes as interactive diagrams – visually displaying dependencies as in a blueprint – and a variety of reports. Role-based users can slice and dice the data to focus on a particular subset, and they can simulate proposed changes to assess the impact: If we upgrade this application or apply that server patch, which business processes will be impacted? If our client load doubles, which applications on which machines will stretch capacity? Or perhaps: Our client load has doubled, and the response time is poor. Where can we offload the excess demand, or do we need to buy some new equipment?

The company started shipping the Troux Blueprinting System in the first quarter of 2002, and has 12 paying customers. A typical deployment starts at $200,000.

Collation: Look but don’t touch!
Collation was co-founded by ceo Robert Roblin, who started his technology career as a product marketing manager at Software Arts, and was most recently vp marketing & sales at Covad – and a user of its Web-based Operations Support suite for operations and “zero-touch” customer provisioning and service. The other co-founder, cto Vinu Sundaresan, led the team that designed that system, one of the largest to date based on J2EE technologies. It was nicely designed, but the reality was constant change: Administrators, developers and the like were continually making helpful changes in the system to accommodate user needs or customer requests –
and those changes were propagating errors through the system, which comprised over 250 servers. They left Covad in 2001 to start Collation, intending to build a system monitoring company. They soon realized, however, that you need to do more than monitor problems to produce actionable information and be able to fix them: You have to figure out what in the run-time infrastructure is causing them.

“We spent our first seven months designing a model,” Roblin says; that is, an empty architectural representation of a J2EE application environment and how its parts interconnect “across tiers” — among themselves and across the three layers of network, systems and applications. Through testing and simulation of this model/tool, dubbed Confignia, Collation has “proved” the model to work and to represent virtually any possible configuration of supported products, Roblin says. The underlying model is an extension of the Distributed Management Task Force’s (DMTF) Common Information Model and JSR77 (an emerging standard model for J2EE components). The model is represented as a schema stored in a relational database (such as Oracle).

For now, the supported products comprise a relatively limited universe — Solaris, J2EE, WebLogic, Netscreen and PIX firewalls, Apache, Cisco routers and switches, Alteon load balancers and Oracle — but one that covers 80 percent of BEA’s installed base, or about 11,000 potential customers, and includes Confignia itself. This set of software can and will be easily extended, and users of Confignia can already add other applications manually.

Meanwhile, the limitation means that Confignia is one of the most automated of the tools. It automatically discovers and maps the topology of J2EE systems, from networking up through system software to applications. It takes less than a day to install and to inspect a typical application installation, using SSH security (guaranteeing that it cannot itself disturb the environment). It can simply poll the entire system and figure out what’s there. It displays the information in an elegant interface, including a nice magnifying-glass-style zoom feature, and charts, tables and dependency diagrams.

Confignia also detects changes: It can’t unroll them, but it can tell you what’s different between two hours ago, when everything was working, and now, when it’s not. It can’t detect changes in, say, the details of a database transaction, but it can notice, for example, the installation of a new firewall or something as minor as changing the

<table>
<thead>
<tr>
<th>COLLATION INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headquarters: Palo Alto, CA</td>
</tr>
<tr>
<td>Founded: January 2001</td>
</tr>
<tr>
<td>Employees: 17</td>
</tr>
<tr>
<td>Revenues: undisclosed</td>
</tr>
<tr>
<td>Number of customers: 4</td>
</tr>
<tr>
<td>Typical enterprise price: $125,000-250,000 for 100-200 servers.</td>
</tr>
<tr>
<td>Funding: $6.3 million from Prism and Worldview</td>
</tr>
<tr>
<td>URL: <a href="http://www.collation.com">www.collation.com</a></td>
</tr>
<tr>
<td>Languages (in addition to English) spoken by the founders: German, Tamil</td>
</tr>
</tbody>
</table>
session timeout on a WebLogic server. While other tools are more granular in detecting what happens within an application, Confignia’s sweet spot is detecting dependencies across tiers—unlike the network administrator who doesn’t realize that his reconfiguration of the load balancer will slow down the inventory updating. (Call it the downside of virtualization: It’s nice to run applications without having to know about the physical infrastructure supporting you, but that physical infrastructure is still there—and if it slows down you will notice!)

As a former user himself, Roblin seems to have a good fix on what users want: They care about security and ease of administration. The system operates from a single server rather than using agents that need to be installed everywhere, and it is designed so that you can see what’s happening but you cannot change it through Confignia. You can also set it up so that any particular user’s view is limited to some parts of the system, again for security. “Someday,” says Roblin, “we’d like to be able to make configuration changes rather than just report them.” But he acknowledges the reality: “We’re still a little company, and we have to prove ourselves. A new company needs to earn the right to touch things.”

Relicore: Hard-core and high-end
Troux doesn’t pretend to be totally automated—and users don’t really trust such claims anyway. But Relicore (which got its name from the concept of reliable core services) takes the automated discovery approach about as far as it can go. Engineer Firdaus Bhathena founded the company at the end of 2000; it was his fourth startup, and the third that he led. After getting both a BS and MS at MIT, he began his career as founding engineer at a start-up that ultimately flopped, and then joined Ray Kurzweil to start Kurzweil Educational Systems. After that, he and a friend won the MIT $50K Entrepreneurship Business Plan competition, and used it to start WebLine Communications, an online customer service and communication software company, in 1996; they sold that to Cisco for $325 million in 1999, and about a year later Bhathena left to start Relicore.

With the new company, he was attempting to address the problems of managing and extracting business value from increasingly dynamic, modular and complex IT environments that he had encountered at customers of WebLine and Cisco. They were “complex, distributed and near impossible for human beings to keep track of,” he recalls. “The only way was manually—Word documents, Excel charts and so forth—and in their heads. These inherent limitations prevented our customers from getting full value from the systems we were building for them.”
What he ended up producing is a system that he regards explicitly as a platform – based on what it calls Dependency Maps (akin to Troux’s blueprints) and a product called Relicore Clarity. Relicore is currently focused on enabling large IT organizations to improve business problem management – i.e. handling “incidents” in real time – and change support, but the information could easily be used in other contexts – most notably security. Everything about a system installation is automatically discovered, analyzed, mapped, tracked, and recorded (in a SQL database, as it happens), with the output then used to inform people trying to understand their systems or change them.

Relicore’s strength is its dependency mapping and tracking tool, which builds a dynamic visual map of an entire IT operation, discovering what’s there, noting dependencies, and thereafter tracking operations and application behavior in real time. It recognizes most major applications and works with core application infrastructure building blocks such as databases (Oracle, Sybase, etc.), application server environments (WebLogic, WebSphere, Microsoft .Net, etc.), Web servers such as Apache and Microsoft IIS, and even DNS servers. It also recognizes what it doesn’t know, and presents the links for a human to label: “This module calls that module and puts the result here. Please give it a name.”

Making maps and watching processes

The main feature of Clarity is that it can monitor processes as well as products. For example, says Bhathena, “You can define a good state and expected behavior, and then the system takes action or alerts you when there are deviations. In normal behavior, for one of our financial services customers [whose name he won’t reveal, but there aren’t many in its class], they connect to twelve different external data feeds, and then send the processed data onward to 180,000 desktops in the Wall Street area. If that doesn’t happen at 4 am every [weekday] morning, there’s a problem. If a process is supposed to be looking at a Web page, we notice if it gets a 404 instead. Likewise, it can apply to security: If something normally connects only to two other boxes, and it suddenly connects to 500 other systems, there’s a problem.” (But for now, at least, Relicore doesn’t market the security aspect, although, says Bhathena, “we have spent a lot of effort on getting these things right.”)

The second area for Relicore is supporting change management. In order to change a system, first you have to know what you’ve got. Relicore doesn’t do automatic roll-
out of updates – which Bhathena considers a well-addressed problem. “But once you've made the change, we can track it and validate it – make sure the right things are talking to the other right things.”

Like Troux and Collation, the company has its own visualization tool, which can display dependency subsets as well as full maps, following Relicore's particular rules of progressive disclosure. "It's not the hairy mess of network topologies, but an organized way to get to the key information that you care about in a dependency map," says Bhathena. "We also have a command-line interface for the real techies. It's not glitzy or glamorous, but nice for getting on with the job."

The company started shipping in the last quarter of 2002, and has five paying customers, some of them rolling the product out to hundreds of servers, he says. A typical initial site deployment takes less than a day and costs about $100,000, with that number going up as the deployment goes enterprise-wide.

**Modeling Data and Schemas: Structured Information**

One of the big challenges for what used to be called MIS has always been to get heterogeneous systems to interact. That challenge has become greater in recent years for two reasons: Companies have been getting larger and more heterogeneous internally, which has led (among other things) to the field of enterprise application integration (EAI, with a focus on syntax: interfaces for interaction and message-passing and coordination).

Meanwhile, the integration now has to happen between as well as within enterprises, given the way the Web and in particular Web services promise that diverse applications can talk to one another. The Internet lets them connect and EAI software ensures that they can communicate, but they aren’t necessarily using the same terminology or data models.

The tools discussed above show dependencies among IT components – infrastructure, applications and the like. But they left the miraculous part to the users, expecting them to resolve the conflicts or reconcile the systems to adjust to changes. They look at large-grained dependencies between applications – who sends data to whom – rather than the dependencies of content or meaning within the data the applications manipulate.
By contrast, the tools covered below help manage dependencies within and across applications by modeling data semantics and schemas. A schema, of course, is passive and knows little about the actual data (or instances). It simply describes the structure of the database and its tables without any data. The data make it real and give it meaning (semantics), the applications make it active...and the users make it alive.

The tools come in two forms:

Contivo and SchemaLogic help users to manage and reconcile data across applications by maintaining a database of data definitions and schemas and transformation rules. Contivo focuses mostly on the data semantics and semi-automated transfor-
nformation rules (for transforming the representation of data from one context to another). SchemaLogic is a collaboration tool built around a database of metadata, designed to support human interaction around the data.

By contrast, Unicorn attempts to model possibly conflicting views of the data and the applications that manipulate them into a single ontology that comprises and reconciles the different schemas. Indeed, just as Unicorn’s tool brings together different schemas, so does the company as a whole bring together the world of legacy IT and the more academic reaches of ontology research. Consider it a bridge to the third section of this issue, on ontologies.

Interestingly, all three companies have a B2B/online market heritage even though they focus on enterprises, and discovered the challenges of semantic incompatibility first hand. Contivo began as a supplier to B2B exchanges and changed its focus last year; the founders have left day-to-day management. In the case of Unicorn, the founder created and sold a B2B company before starting Unicorn. At SchemaLogic, meanwhile, cofounder Trevor Traina had earlier founded Compare.net, a B2C site devoted to finding similar products in different contexts.

Because these tools “understand” the meaning of data structures and can map them from one context to another, they are well suited to be players in the service grids John Hagel and John Seely Brown described in these pages (see RELEASE 1.0, DECEMBER 2002). (The issue mentions Contivo as one example.)

**Contivo: Translating taxonomies**

Contivo starts with the seemingly simple task of translating application/data elements from one environment to the other. Started in 1998 to sell to exchanges and B2B projects such as ForestExpress (for the wood, paper and forest products industry) and Trade Ranger (petrochemicals), it changed its focus last year and raised an additional round of $8 million in January 2003. The business plan predicts profitability early in 2004.

Now focused primarily on enterprise customers, Contivo talks about “enterprise integration competency center” — a notion it has sold to Hewlett-Packard among others. HP is a great reference account — high-profile and rich in examples, and undergoing a change dramatic enough that it has been willing to adopt the technology wholesale, with an “integration competence center” serving the newly combined company. HP has an unlimited license to use Contivo’s technology, and the company
has already trained over 75 HP systems analysts in the US and Europe to date, with more to come. HP has at least eight different runtime engines, and Compaq has at least six; Contivo provides them a single modeling and design environment to rationalize all their different environments. Other customers include Hitachi, which used it to migrate from a legacy EDI (Electronic Data Interchange, a longtime industry standard) system to webMethods, without disturbing those trading partners who remained on EDI.

Contivo also works with standards organizations such as RosettaNet and the Open Application Group; cto Dave Hollander is co-chair of both the Web Services Architecture and XML Schemas working groups for the W3C.

Currently, the primary value – in ROI to the customers and revenues to Contivo – is within corporations trying to get their applications to interoperate. “The challenge is the number of data buckets that need to be integrated,” says ceo Larry Lenhart. “From one to another is not so hard. But over time, they accumulate. We give you the ability to record and reapply these [transformation/data integration] decisions as transformation rules.” Once a user defines the mapping between two data elements, that particular capability can be used over and over, or modified as either data set changes.

The product comes in two components, the Contivo Analyst and the Enterprise Integration Modeling (EIM) Server. The Analyst lets users specify their data and synonyms, and imports schemas and data definitions into the EIM Server. The EIM Server in turn stores a dictionary of concepts and a thesaurus of synonyms, including the schemas and data definitions used in all the different application environments any particular enterprise supports, in a standard relational database (Oracle or SQL Server). Contivo ships the EIM Server pre-populated with 4,600 business concepts drawn from the Open Application Group’s and other groups’ business object definitions, for functions such as procurement, payments, and inventory management. Customers can also order industry-specific data dictionaries for fields such as insurance.

The server also contains transformation rules, so as to map data from one application term to another. These can be created by a developer with the Analyst, or (like the dictionaries) various standard ones can be purchased. The transformation rules describe how to move, format and transform data for a specific source/target combi-
nation. For example, the semantic models for an SAP purchase order and a RosettaNet purchase order are the same whether SAP is the source or the target, but the rules to transform one into the other are inverse. The rules deal both with the specifics of the data semantics and with the more general issues of moving from, say, EDI formats to WebLogic flat files or from Oracle to XML. For a fee starting at around $4,000, Contivo’s MapFactory will develop customized data-transformation maps for customers too busy to do so themselves.

For practical purposes, many of the terms and rules are duplicates or specializations of more general terms, so they are stored in a four-level hierarchy that’s invisible to the user. That is, once you’ve defined a customer, defining a “preferred customer” requires only a little extra data. Likewise, a Hertz customer and a United Airlines customer have many fields in common, but Hertz customers have car preferences whereas United customers have seat and (at least in the old days!) meal preferences.

What differentiates Contivo from its competition is that it’s a design-time tool that supports multiple different runtimes rather than a real-time mapping engine such as Mercator’s, a well-known player in that space. Thus it’s best suited for internal or partner use, where you want persistent interfaces, rather than in a more open environment where you never know what your applications will be talking to next. Developers use it to develop and compile interfaces between applications, which then execute without further reference to Contivo’s EIM Server.

**SchemaLogic: Schemas by subscription**

SchemaLogic was founded two years ago by Andrei Ovchinnikov, Trevor Traina and Breanna Anderson, all three previously at Microsoft (hence the location in Redmond). Traina had sold his company, Compare.net, to Microsoft. All have considerable experience with taxonomies and schemas, before and during their time at Microsoft. Ovchinnikov ran several taxonomy development and management projects across Microsoft’s many groups and divisions, while Traina’s Compare.net used taxonomies extensively to classify and compare consumer products for sale on a variety of Websites. Anderson was one of the original architects behind MSNBC and several other large Web properties of Microsoft, and developed Interpress, a template-driven publishing tool.

In many ways SchemaLogic is similar to Contivo in its basic mission, but it operates very differently, as a collaboration environment that handles the human as well as the technical issues in data and schema reconciliation across environments.
SchemaLogic’s SchemaServer also works with a broader range of data types including documents and Web pages, whereas Contivo focuses on structured data used by database-oriented applications. The underlying notion at Contivo is to define data and interactions centrally and then put the appropriate interfaces into the applications, almost automatically; the idea at SchemaLogic is to manage the development of the schemas in the first place, bringing the data and schema “owners” as well as the programmers into the process.

All data and schemas belong to someone in the SchemaLogic approach, and users as well as developers have a say. When a schema is created or modified – after approval by the parties involved (subject matter experts, business users, librarians/taxonomists, system administrators et al.) – the changes are propagated through to the affected applications.

The SchemaServer manages extensive information about a company’s data and content, including everything from data definitions and schemas for any structured data, including not just database data but also documents and Web pages or even press releases. Its “goal” is to get people to agree on data definitions and schemas, and to the extent that that doesn’t work, it supports them in developing reusable transformation rules between them.

All these items are tagged by who “owns” them, who can change them, who can veto changes, and who must simply be notified of a change. The system maintains appropriate relationships, so that someone who owns an entire vocabulary has the same privileges over all its constituent or related parts. The UI essentially presents the schema information it has, allows users to modify it according to specified roles/privileges, and makes sure both the changes and the information about the changes is propagated properly.

The system is more focused on trying to keep a company’s data structures and schemas in synch, rather than automatically reconciling them. However, it can import substantial amounts of metadata about each system, especially from partner-company tools such as Stellent, Interwoven, BEA and IBM. “We don’t sell languages or push any particular system integration strategy,” says cofounder Anderson. “Our piece of the puzzle is to provide the flexible but precisely actionable definitions that can be used to drive configuration, mapping and synchronization processes in the network of applications.” The output – once all parties have played their roles – can
include everything from schema definitions in a variety of languages to a configuration file that identifies matching data across dissimilar systems and even languages.

SchemaLogic’s assumption is that tools can manage the content, but that people need to drive the tools. Although it’s actually quite automated, it ensures consensus before it takes any actions.

**Unicorn: Outside the normal taxonomy**

Zvi Schreiber, founder of Unicorn, previously founded Tradeum (see **Release 1.0, September 1999**). There he learned the hard way the amazing disconnects between different users’ assumptions about their data and its meaning (even though it was easy to build a transaction engine to match bids and offers). “At Tradeum, we created a very fancy matching engine, based on matching specs, but in the end different buyers and sellers were talking about different things,” he says. It wasn’t just that Juan used Widget while Alice used Juidget. Nor was it simple software incompatibilities, such as translating from Java to .Net. It was things you could discover only by understanding two cultures – as he did, being English-born but raised in Israel. (Moreover, Schreiber points out, “I moved with Tradeum to California and I now spend half my time in New York – which as you know are not one but two extra cultures.”)

So rather than build taxonomies and rules to map data from one schema to another, Unicorn builds ontologies, or descriptions of schemas and application logic. That is, it builds one ontology per client: “The key value is in having a central ontology rather than ‘ontologies.’ The ontology captures a common model of the business (or common business world-view) and maps physical data schemas onto itself. It holds the objective business meaning (or semantics) of all the data and the applications in one place.

“Only a human being really knows what the data means,” continues Schreiber. “The data descriptions we use are not automatic; they’re ‘computer-aided’: We suggest the likely match and let the user confirm. We catalogue all the data, and then we map how they are related. Then we use the metadata to provide services: data management, data integration and data quality.

“The emphasis is on tangible value,” he adds, repeating a popular refrain. “We can actually generate code to actively achieve data management, data integration and data quality in contrast to passive metadata repositories.”
Unicorn can parse most common database schemas and data-modeling tools. The underlying technology it uses to represent the schemas is a home-built engine based on the emerging W3C standard, Ontology Web Language (OWL). It captures the meaning of data and maps it to an ontology: “The core technology is actually more in the mapping than in the ontology,” says Schreiber. The value thereafter, for the customer, is that you can “execute” the ontology; if you change something, the changes ripple appropriately through the system and redefine the resulting schema appropriately, while the engine detects changes and then runs rules to keep everything consistent.

“We don’t produce ‘code,’” says frequent-flyer Schreiber, “but we can automatically can create and modify the translation scripts that keep different data schemas talking to each other. For example, say a business rule changes – the rules for calculating mileage for a customer to reach Platinum status. Now every place that matters – from a database that talks about the number of miles to a database that talks about Platinum status – needs to reflect this new logic. With our tool, you make the change into our model [ontology], and all the related schemas and scripts are updated automatically. If two airlines merge, you may need to manage both rulesets independently, while merging the two customer databases. Capturing business rules centrally and applying the right ones locally is key to automatically translating between semantically different systems."

Do you know globally what you know locally?
The user problems Unicorn encounters and addresses are incredibly simple, and incredibly deep. For example, one large electronics manufacturer has five different systems involved with scheduling the flow of goods: an ERP system, an advanced planning and scheduling suite, plus three others in the manufacturing operations, each with different semantics. The company was making mistakes; goods weren’t where they should be. On one occasion trucks came to a warehouse to pick up some components for Compaq that weren’t there, a mistake that cost $10 million because the price had dropped by the time they were found the following quarter. “That [supplier] company lost well over $100 million a year because of data quality issues,” says Schreiber.

As Unicorn went through just two of the five systems and modeled their data side by side, it found numerous discrepancies. One significant one that had escaped detection for years was the definition of the working week. For one system it began at 8 am Monday Pacific Standard Time; for some of the manufacturing operations’ systems, it began more than 30 hours earlier, on Sunday morning in Taiwan.
“Whatever Taiwan and Israel had been working on for a day and a half, belonged to the previous week in California,” says Schreiber. “We did a fairly sophisticated semantic mapping to understand how things related, and then we translated from one to another. You don’t need to make everything the same; you just need to know how to translate properly.”

The unnamed Fortune 100 client (whom we talked with) is enthusiastic: “It’s much easier to maintain a model than to maintain a SQL script. When you change something, the script is rewritten automatically. The amount of time you spend developing the model vs. writing the script manually for a point-to-point translation took about the same time – about three days. But afterwards, on maintenance, we estimate it takes only half the time. Beyond that, there are fewer errors. And it’s scalable: You can add another comparison [i.e. a third system] to the same model for very little extra effort.”

But, Schreiber points out, it’s not always a question of multiple systems. Sometimes it’s enough of a challenge to figure out what’s in a single one. In the case of another client (whom we also interviewed), he says, “They had an operational data store that goes back 20 years. No one really understood it. The problem in this case was how to update the system to offer more flexible pricing, take advantage of yield management, and basically join the modern world where rates go up and down according to a variety of criteria. [See RELEASE 1.0, FEBRUARY 1989, ON YIELD MANAGEMENT] They had all this data and all these scripts, but they couldn’t update the system because no one knew what was in it. The business logic was all hidden, so it was dangerous to change anything without knowing what it might affect.”

According to a manager at this client, “The programmers and technical people said we already have this information. . .and so they did, in COBOL copybooks and early versions of PowerDesigner (from Powersoft, since acquired by Sybase), and IMS hierarchies. It was in computer language rather than meaningful language. But afterwards, they actually thought it was worthwhile to have the model.” Whereas the business analysts appreciated it almost from the start, and were eager to have an easier way of finding out what kind of data they could get and then how to design queries, reports and analyses without having to beg for help from a programmer. Meanwhile, the programmers can now update the applications with flexible, explicit rules both developed and visible in a central ontology.
Modeling the World: Why? And How?

The fundamental question people are trying to solve with ontologies is not to create better database queries, nor even to describe the world better (though those are useful capabilities), but to be able to switch from one worldview to another, whether across systems, or across time. What they really want to do is understand the systems they have, relate them to their actual business conditions, and then be able to modify them automatically as those conditions change. These are simple problems. But now magnify these nuances and complexities by thousands, and then imagine a merger, a divisional reorganization or the like...and imagine trying to make sense of it all.

Technically, this is called “semantic integration”, which means that you can meaningfully combine data from different data sources. There are a lot of different levels to that. The easiest is just getting data into a common data format: XML or ODBC or any standard database. Then there are the data-mapping tools we described above. And finally, there’s a full-fledged ontology, which can model the complex relationships implemented in applications that do something, as well as just schemas that map the data. The power of an ontology is to generate new information beyond what was put in, by reasoning or using short-cut techniques such as inheritance or specialized rules.

The toolsets generally come in four parts (though which plays the central role varies from company to company):

- **The development tool.** This is the tool that you use to build an ontology. It can have a GUI. It may import data and schemas from other formats. And it may do error-checking or light inferencing to test the model. In any case, it helps a non-technical domain expert write ontologies without knowing the underlying implementation details, just as a Web-design tool lets a lay person write HTML.

- **The ontology engine.** This executes the ontology, building a complex model and generating relationships dynamically by reasoning about the information it knows. In general, ontologies focus on representation of complexity rather than performance or implementation efficiency.

- **The broker/integrator.** This component can translate an ontology back and forth into different IT environments, ideally providing the integration between different application environments, or just producing a single application that implements the model and transactions defined by the ontology. In theory, if the ontology engine were powerful and fast enough, you wouldn’t need to go back into the traditional IT environment, but for most applications other than design
or simulations, the broker enables the ontology to interact with the real world of massive data sets (everything from customer databases to financial records and factory processes). In essence, the broker lets the engine do what it is good at – reasoning – and lets IT do what it is good at – processing transactions.

- **A query tool.** One benefit of an ontology is to represent a domain in a way understandable to humans. The query tool lets humans interact with it.

Like it or not, we need to use that awkward word ontology (see also Release 1.0, January 2003). It’s awkward because it is commonly used in at least three ways, to refer to:

- the study of being, in philosophy;
- (loosely) a thesaurus or (passive) concept map, usually for information retrieval or in some semantic Web contexts;
- a structured model, complete with taxonomies of both data and relationships and arbitrarily complex dependencies and logical inferences enabled, in techy/academic IT circles. That’s the sense in which we use it here.

An ontology, then, is an active model that contains a variety of data structures and some way of propagating changes through itself. It can comprise a host of things: taxonomies of data objects; taxonomies of relationships or typed links (often expressed as verb phrases), from “is associated with” to “is a kind of” to “contains” or “produces” or “consumes” or even “enjoys” or “prefers” or “burns.” Those relationships can usually be modeled or represented by combining other more elemental components, or through applications that implement (for example) all the things that a customer can do or can have done to her and her account. (Another example: Burning is a specific kind of destruction; it is also a chemical process. Which representation you use depends on the context.)

An ontology is active: An ontology is not an application in the traditional sense, but it can execute logic to make inferences about unstated facts, and navigate graphs to represent complex webs of relationships. It can reason about new or changed information, and draw conclusions that can flag inconsistencies or even drive a tool to generate code. “If what you get out of it is only what you put into it, what’s the point?” asks Ontology Works co-founder Joshua Engel rhetorically (see page 31).

The relationships in an ontology can be arbitrarily complex, and it often takes many schemas and applications to implement them in legacy systems. But those relationships are often not clear or even visible by the time they get implemented in code. The trick is to make the model intelligible and actionable — able to reason about itself and to produce working code to implement its meaning.

There’s no canonical form for an ontology, as there is for a relational database, with its schemas of tables and joins, or a taxonomy, with its powerful hierarchical inheritance mechanisms. An ontology comprises all these and more, reflecting the complexity of the real world. Ideally, it’s close enough to reality for lay people to use effectively. The practical challenge for IT is to build ontologies that can represent complex relationships and still work fast enough to be useful in operation. (A poorly designed ontology can spend days reasoning about irrelevant facts in search of the answer to a simple question.)
### ONTO INSIDE!

<table>
<thead>
<tr>
<th>Company</th>
<th>Development tool</th>
<th>Engine</th>
<th>Broker</th>
<th>Query tool, other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unicorn</td>
<td>Unicorn Workbench -- ontologies, maps, schemas</td>
<td>Unicorn Server infers data transformations, description logic</td>
<td>Into and out of standard IT schemas</td>
<td>Generates SQL, other queries</td>
</tr>
<tr>
<td>Cyc</td>
<td>Learning dialogues</td>
<td>Cyc itself, first-order logic</td>
<td></td>
<td>Browser-based</td>
</tr>
<tr>
<td>enLeague</td>
<td>Own GUI or third-party</td>
<td>Uses Cerebra or Prolog or others</td>
<td>Semantic Integration Broker</td>
<td>Semantic Browser</td>
</tr>
<tr>
<td>Network Inference</td>
<td>Cerebra Construct (Visio plug-in)</td>
<td>Cerebra (based on FaCT), OWL-native, description logic</td>
<td>Cerebra Central</td>
<td></td>
</tr>
<tr>
<td>Ontology Works</td>
<td>IODE GUI that edits, checks consistency</td>
<td>Well-Founded Semantics, akin to first-order logic</td>
<td>IODE Mediator</td>
<td>IODE, ODBC tools</td>
</tr>
<tr>
<td>Ontoprise</td>
<td>OntoEdit, frame logic</td>
<td>OntoBroker, frame logic</td>
<td>OntoBroker, connectors to Oracle, DB2, MS SQL...</td>
<td>SemanticMiner, OntoOffice</td>
</tr>
</tbody>
</table>

This is an attempt to put each vendor’s product suite into a common framework without obliterating the distinctions each one wants to highlight; sounds like a good use for an ontology. Precisely how the functions are divided up and what weight they are given varies from company to company. In the table above, we have highlighted the key products from each company. (It’s a sign of the market’s immaturity that when we talked with them, many of the companies mentioned their actual product names only as an afterthought.)

The power of an ontology lies in its ability to model complexity and infer implicit information, whereas the power of a traditional database lies in its speed and scalability. Cyc is epitome of an ontology; it uses powerful first-order logic (SEE PAGE 23), holds huge volumes of information, and favors completeness and expressiveness over performance.

Ontology Works, likewise, has a powerful first-order logic ontology engine and focuses on complex modeling, and it includes a broker to produce application code for better performance. It is strong in fields such as aerospace and biotech and homeland security.

By contrast, Network Inference and Ontoprise use frame or description logic (a less powerful approach that focuses on data relationships rather than reasoning) to make a trade-off towards better performance and somewhat less complexity. Ontoprise has the most mature and balanced toolset, with over 50 commercial installations, many of them in industry/operations rather than traditional IT. However, it also is gaining presence in front offices with its OntoOffice query tools. Network Inference, a newer company, is still mostly focused on scientific or design-oriented applications, but it is also being used (for example) by enLeague as part of its Semantic Broker in a CRM application (SEE ALSO THE COVERAGE OF BABY CARE LINK IN RELEASE 1.0, JANUARY 2003).

And finally, Unicorn and enLeague focus on more traditional commercial applications, where the ontology is used primarily to drive database applications and the brokers play a key role in translating the ontology back and forth. The actual data is held in databases rather than manipulated by the ontology itself. (We put Unicorn in the “modeling data” section because it positions itself squarely in the IT world, whereas enLeague focuses on ontologies in its positioning.)

All of them are watching the W3C standards efforts with interest and following their guidelines, but their practical implementations extend way beyond what the W3C has canonized. (Several of these companies’ ceos will be participating in a showcase panel at PC Forum.)
Cycorp: The model for them all
Doug Lenat’s Cyc is the grandparent of large-scale ontologies, a project the longtime AI researcher and his team have been working on since 1984, to the tune of 700 person-years. We first wrote about it in March 1986. The overall idea is an all-singing, all-dancing ontology of the universe: It is to search engines and directories (or the Semantic Web) what Ted Nelson’s Xanadu was to the current Web. But it keeps growing, and while it is not yet IPO material, it is employee-owned and self-sustaining on the basis of R&D grants and some contracts.

Last year it reached a significant turning point: Instead of being a repository into which people put knowledge, it now knows enough to ask reasonably relevant questions to enlarge its understanding. In theory, Cyc should have been a kind of bottom-up ontology version of the Open Directory Project (see release 1.0, January 2003), but entering information into it was too complicated. Two years ago, Cycorp launched OpenCyc to foster that very thing. “We’ve moved from adding knowledge by virtual brain surgery,” says Lenat, “to adding knowledge by tutoring, and that’s something we ought to be able to enlist many other people to help with.” (One approach might be drawn from the world of multiplayer games; why not have users construct the real world rather than artificial ones?)
Overall, it will be interesting to see how that works out. There is not and cannot be one single worldview; there will always be multiple taxonomies and ontologies that can work. Instead we need to figure out how to translate from one to another (as illustrated elsewhere in this issue). Lenat and his colleague Ramanathan Guha recognized this early on, and since 1988 Cyc has organized its knowledge into contexts or “micro-theories” that differ by time, place, level of granularity, and so on. An assertion can be true in one Cyc context and false in another (though this feature must be used with care!).

“There’s an almost infinite number of unstated assumptions in everything you do,” says Lenat. “So we map out shared-assumption spaces and let the system ignore things beyond them, so that it doesn’t get caught up in irrelevant questions and can reason efficiently.”

Technically, Cyc began as a taxonomy with inheritance, along with IF-THEN rules. But it was more and more of a strain to shoehorn into this representation the sorts of knowledge that comprise common sense: sentences involving OR, NOT, beliefs, and so on. Ultimately, in the late 1980’s, Lenat and his team moved grudgingly to formal logic to represent knowledge in Cyc. They did this with considerable trepidation, since reasoning over a large set of logical assertions is combinatorially explosive. To recoup the lost efficiency, Cycorp developed the context mechanism described above, and added many special-purpose reasoning modules to handle commonly occurring types of problems.

By 1989, Cyc had 20 specialized inference modules; now it has 525. These are shorthand reasoning modules that make it possible to represent complex relationships or reasoning sequences without going back to first principles. For example, one of them re-represents facts as a graph, then does graph searching which, if it works, is much more efficient than doing the same reasoning by logical deduction.

“For example,” says Lenat, “to disambiguate some English sentence, Cyc might need to know whether a bird is opaque. Now you don’t want it to call on a general theorem prover to come with a 10-step proof in two hours; you want it to search a graph and inherit that information from ‘Tangible Object’ (which as a default is opaque), or you want it to examine the archetypical bird it knows about and ask whether that bird is opaque, or... you get the idea: Use some specialized means to get the answer, not a slow theorem prover.”
Or suppose you own a car. Do you own the lugnuts on its front left tire, too? Of course you do, but how do you know the answer to that instantly, rather than through slow pondering?

The modules/rules deal with a variety of relationships, and they can “specialize” just as classes of tangible things do. For example, there are rules about size and fitting: people can fit into seats, and people can fit into houses, and certain sizes of clothes, things can fit through doors, and so forth. An airplane overhead bin can hold a certain amount of luggage of requisite dimensions. All these are instances of the Containment Reasoning module.

Other modules deal with kinship relationships, reasoning about pieces of time and temporal relationships and causality, reasoning about pieces of space, pathways, and travel along paths, and so on. There are some relationships that transfer to the components of the entities, and others that don’t. For example, you may love a country but not all its citizens.

Simply classifying tens of thousands of those kinds of relationships and how they “unbundle” created another jump in the system’s capabilities and power. “These things work on the basis of heuristic rules of good guessing, rather than logic,” says Lenat. “In all cases, the system would give exactly the same answers if you removed all 525 modules, but it would take a lot longer. Think ‘heat death of the universe’ longer.”

Cyc’s ontogeny

Cyc started at a time when American industry was afraid of Japan’s newly launched Fifth-Generation computer project and money flowed freely for long-term, high-risk, high-payoff research. It began as a project of the MCC research consortium in Austin, TX, funded by a dozen large U.S. companies including DEC, Kodak, TI and Westinghouse, and later on Apple and Microsoft.

Cycorp spun out of MCC in 1995 and is beginning to catch some commercial attention, though most of its revenues are from government contracts. “I’m willing to sacrifice a little of the purity of the dream of building the world’s first true AI to get revenues, but fortunately I only have to sacrifice a little,” says Lenat. “It’s like tacking a sailboat into the wind of financial constraints. We have to angle our sail to catch some of that wind; we’re going about half in the direction we want, and about half with the wind.”
Overall Cycorp has 66 people and about a dozen government contracts, half of them with DARPA including a $9.8-million contract for a system that could model terrorist behavior (but none directly with the Total Information Awareness office).

The current commercial customers include Streamsage as a partner in a contract with National Public Radio to help in annotating transcripts with metadata. The metadata are then used so that transcripts can be searched for references that may not be easily findable – for example, when vague references are made to people or things mentioned earlier in a show, or phrases such as “what happened last week.”

Another client is GlaxoSmithKline, which is using Cyc in a very specific way – as a tool to manage the different meanings of terms in different subfields of medicine, different countries, different companies, different years, and so on. Just as the US and the UK are “divided by a common language” (e.g. boot vs. trunk of a car, lorry vs. truck, sweet vs. candy), so are various medical fields. Cyc contains knowledge about the relevant terms in medicine and is used by thousands of GSK employees daily to as they search and classify documents.

These are nice examples of the kind of specificity that Cyc can provide – once it has acquired the relevant knowledge. But they also illustrate the commercial challenges Cyc faces: “We couldn’t sustain the entire company on this kind of application,” Lenat acknowledges. “It’s a stepping stone.” In fact, it’s like using the automatic screwdriver on a factory assembly line to repair your bicycle – useful and cost-effective in its own way, but not the optimal use of the entire factory. For instance, the company had a number of contracts in the health-insurance sector, but in the end the clients didn’t want to pay the overhead of Cyc’s extensive knowledge base; Lenat calls it “a tendency toward short-sighted over-specialization.”

Nonetheless, that 700-person-year overhead devoted to building Cyc’s ontology at a high level of generalization is now beginning to pay off because the system can absorb new knowledge more easily, but the path has been long (though no longer than it takes to develop a college student from an infant). Cyc now has the ability to engage in clarifying dialogue with a lay user. That is not entirely an abstract ability (though it depends on underlying logic): It requires the system to know enough to ask useful questions. For example, if someone tells Cyc “anthrax can kill human beings,” it can ask a range of useful clarifying questions: How long does it take? How does it kill humans? Does it kill other animals as well (which kinds)? Are there ways to counteract the killing process?
**Current ontology**

While many commercial customers may not want to pay the entire cost of an ontological factory to run one small line, Cyc’s breadth is compelling to another kind of customer, one that does want to model and track the entire world in order to detect the terrorists within it. Cyc is intriguing a variety of government officials, many of them in DARPA and three-letter agencies, with its ability to model behavior and scenarios and to extend its reach across domains previously thought irrelevant: everything from aviation practices to car rentals, human living arrangements, geographical considerations, weaponry and its uses, and the use of non-weapons to inflict damage in similar ways. Cyc has some common sense, but it also is more able to see things that a human might miss out of mistaken assumptions. (Indeed, it might be interesting to see whether there are incorrect assumptions or defaults in Cyc that can be removed more easily than they could be removed from humans – for example, that airplanes are not weapons.)

Those who care about privacy can note that Cyc concerns classes; the privacy issues arise only when Cyc (or other models) get hooked up to instance data.

**Network Inference: Going commercial**

Network Inference, a British company recently transplanted from Manchester University to more commercial surroundings in northwest London, offers an inference engine that implements the description logic level of OWL. Network Inference’s own founder/developers were among those developing these standards, and the company has good ties in the academic and EU research community. Advisor Ian Horrocks, a senior lecturer at Manchester University, developed the FaCT ontology engine on which Cerebra, their product, is based.

The company now also has an impressive management team of practical-minded people with commercial experience, including ceo Jon Matonis, formerly ceo of Hush Communications (secure Internet communication) and before that director of financial markets for VeriSign; and vp engineering Jack Berkowitz, formerly vp product operations for Reef.

Cerebra follows the W3C and academic standards, but it is also fast; most ontological engines are slow and can’t handle real-world complexities in the time required. It achieves this by using description logic, which lacks some of the expressiveness of, say, Cyc’s deep reasoning, but makes up for it in speed and performance for many applications. And, just as a commercial relational database offers more than just
schemas, Cerebra has facilities for stability, traceability and performance. It also offers a plug-in to Visio, called Cerebra Construct (developed with technology from Semtation in Germany), that lets users model and test their ontologies graphically.

Like many other companies with a research heritage, NI is working with partners to reach the commercial market. “Our OEM plans are progressing more rapidly than direct sales,” says Matonis. In addition to enLeague (below), the company is working with IBM’s Watson Research Labs. “We have more people at IBM dealing with us than we have in our own company,” says Matonis. This may indicate that customers would rather “own” their own ontology than share part of a larger one, such as Cyc. In the end, “Ontologies will be built from the bottom up, just like trust networks,” says Matonis, reflecting his experience at VeriSign.

On the direct side, NI is selling to Qinetiq, the “commercial arm” of the UK Ministry of Defense Research Agency, and adventurous startups such as CSTlink (see RELEASE 1.0, JANUARY 2003).

Network Inference’s major challenge will be to grow while waiting for a few of its customers to succeed. Fortunately it has $4 million in funding from Nokia Venture Partners, which should carry the 17-person company through to 2004, with revenues starting this year.

**enLeague: Fizzy logic**

EnLeague is a startup in Fizzion, Coca-Cola’s business accelerator. Fizzion was created in late 2000 by Coke ceo Doug Daft (in partnership with the Atlanta Tech Development Center, which gave the world Mindspring/Earthlink), to explore what use Coke could make of e-marketplaces and customer self-service portals. Fizzion was led by Chris Lowe, now head of marketing at Coke, and consulted to by Brian Dyson, who at the time was retired from Coke and has now returned as vice chairman. His involvement was key in getting high-level attention for Fizzion and enLeague at Coke.

The enLeague team rapidly discovered that Coke had an ontological challenge: It had to struggle to define who its customers were. Coca-Cola has five divisions, 88 bottlers in North America and over 200 worldwide, a string of global accounts, fran-
chisees, 20-plus brands, and other complex relationships to manage across the globe.
(And you thought Caffeine-Free Diet Coke was complex?)

That complexity was highlighted by the complexities of managing promotions and
deliveries around events such as the 2002 World Cup. During the two weeks of that
event, the company and one of its largest customers, convenience chain 7-Eleven,
had 500 people on the case, relying on e-mail and Powerpoints and Excel spread-
sheets to handle data from Coke’s vast array of systems and data warehouses and to
manage a broad range of ad hoc activities with a variety of partners around the
world. Never again! everyone vowed.

The challenge of dealing with all that enticed Dwight Lodge to become enLeague ceo
in April 2002. Like Srinija Srinivasan at Yahoo! (see RELEASE 1.0, JANUARY 2003),
Dwight Lodge got his grounding in ontology at Cycorp, where he was ceo from
September 2000 to September 2001. But before that he was no academic: He had
worked as a VC and buyout man in New York, where among other things he bought
and rolled up a group of telephone-billing service providers.

At Coke, there seemed to be no easy way to consolidate all the different views of the
data proliferating throughout the company and its partners. “What Web services
currently offer dynamic binding?” Lodge asks rhetorically.

Lodge extended enLeague’s initial Prolog/Java approach to comprise
other inference engines such as Cerebra from Network Inference (see PAGE 27). Using DARPA’s Agent Mark-up Language (DAML), enLeague’s
Semantic Integration Broker represents and mediates among Coke’s
applications, databases, and business processes, expressed as a series of
related ontologies. It also communicates with the “IT world” of legacy
applications, J2EE, WebSphere, plus a beta version of IBM’s Experanto, a
data management solution that sits on top of structured and unstruc-
tured data and eases it into a single DB2 view. “Experanto forces every-
one to a single view of the data,” says Lodge. “Japan looked at it and said,
‘It doesn’t work for us.’ Europe said the same thing. The broker lets each
group keep its own system and still communicate effectively.”

With the strengths of Network Inference’s Cerebra and DB2 allocated
appropriately, Lodge believes the system will scale to support even an
enterprise as large as Coke. “We need something robust,” he says. “Look at Coke’s vol-
ume of instance data: 16 million different partners out there handle Coke. “

<table>
<thead>
<tr>
<th>ENLEAGUE INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headquarters: Atlanta, GA</td>
</tr>
<tr>
<td>Founded: September 2000</td>
</tr>
<tr>
<td>Employees: 15</td>
</tr>
<tr>
<td>Revenues: none</td>
</tr>
<tr>
<td>Number of customers: in beta with a number of customers</td>
</tr>
<tr>
<td>Typical enterprise price: $250,000</td>
</tr>
<tr>
<td>Funding: $3.0 million from private investors, Fizzion/The Coca-Cola Company</td>
</tr>
<tr>
<td>URL: <a href="http://www.enleague.com">www.enleague.com</a></td>
</tr>
<tr>
<td>Languages (in addition to English) spoken by the founders: none</td>
</tr>
</tbody>
</table>
The unbearable lightness of billing

The new system is now being tested in enLeague’s labs. The OIL+DAML system represents Coke’s distributor net as a 3D structure – location, organization structures, and (jointly) products and promotions. “Now companies such as Coke can have multiple views of the same data, based on geography, who you are, or how you prefer to buy,” says Lodge.

Coke’s customers come in many shapes and sizes and at various levels in the distribution chain. They are not just bottlers but also franchisors and franchisees with multiple trade names and a variety of ownership structures, such as a local entrepreneur who might own franchises that own and operate a White Castle, a Motel 66, and a gas station. Some buy centrally; some locally. Some sell by the bottle; others operate fountains.

In each case, just who is the actual customer? What contract are they eligible to buy under and which bottler last serviced the account? Are there commissions due, or discounts? How are the amounts aggregated to meet negotiated contract levels?

EnLeague’s number-one priority with Coca-Cola is creating a semantically integrated customer master file for large organizations – but a dynamic one. When, for example, a franchise changes hands, the proper relationships can be reinstated automatically: Where and from whom does the changed entity now buy its Coke? How are other partner relationships affected? One or two such changes requires a good lawyer. Hundreds of them per year require...an ontological representation that can recalculate the relationships and dependencies automatically.

Once that is done, says Lodge, there’s the product master file: not just different kinds of products, with formulas varying from country to country, but also different product packaging and other factors. It’s not just a question of pricing and billing, but also of production efficiencies, volumes and logistics.

While this is just the beginning of a five-year project at Coke, Lodge says, his main job is to help the company extend its reach beyond Coke. There are many other companies facing similar commercial complexities as Coke, including the telecom companies Loge is familiar with. In banking and finance, one prospect that enLeague is talking with is an insurance broker that likewise wants a better handle on its customers, for products ranging from straight commercial insurance to D&O coverage (hot right now) and 401Ks and money management. Within each cus-
tomer different people and groups are prospects for these various services, and in the middle are advisors, gatekeepers, agents and lawyers.

In addition, enLeague is looking at verticals such as health care and biotech. Founder and director Dan Pompilio earlier co-founded ActaMed, which was acquired by Healtheon (now WebMD), and three of its other top executives also have health-care backgrounds, so the company has a good connection into that market. It has just announced plans to acquire Killdara, an XML messaging platform company in Ottawa, Canada, that has a deal with EDS to implement the Canadian government’s Smart Systems for Health initiative nationwide. EnLeague’s Semantic Integration capabilities will allow Smart Systems to intelligently manage the diverse Web services and XML infrastructure it’s building to integrate hospitals, labs, clinics and physicians across the country. Sounds like another non-Coke challenge!

And finally, like its peers, enLeague is talking with the US government about homeland security applications.

**Ontology Works: Total ontology awareness**

Ontology Works, as its name suggests, is squarely focused on building high-level, expressive ontologies. Four years old, it has developed a tool set — Integrated Ontology Development Environment (IODE) — designed to make it easier to develop an ontology and QC it — before trying to run it, only to find out that it has inconsistencies and doesn’t “work.” To beat the trade-off between expressiveness and performance, it uses the ontology to generate more standard software components to implement the ontology’s models.

OWI uses Well-Founded Semantics, a first-order logic with a few extra features that improve performance, such as the ability to express negative statements (which saves a lot of unnecessary reasoning), and its own proprietary data structure. Says CEO Mark Diggs, “We have the business and domain experts building the model, and then it generates software components for the IT people to implement in a higher-performance environment.” Those components include DB schemas, API’s, documentation, DAML and XML.

The company is based not in California but in Odenton, MD — within the East Coast hotbed for bio-engineering as well as government contracting. Three of the company’s founders previously worked in technical positions at the National Security Agency. The fourth, Joshua Engel, author of *Programming for the Java Virtual*
Machine, worked for 10 years for military contractor SAIC. Smart enough to realize they needed a businessman to run the place, they brought in Diggs in the summer of 2000. Diggs previously founded and took public an Arkansas-based systems integrator called Bright Star. (Last year’s PC Forum speaker Wes Clark is on the board.)

“We don’t want to create a Cecil B. DeMille company – a cast of thousands and years in the making,” he says. “We want to be agile and focused on ontology tools, and partner with people like enLeague or IBM. Our technology provides them with a strategic differentiator that will result in new revenues streams for them...and in turn for us.” The company has already worked with several major companies and government agencies on DOD-related contracts it can’t discuss, but it does have some commercial customers, especially in the fields of biotech (genomics and proteomics).

Another promising field for OWI is health care. The company is currently working with a major health insurance company on ICD9 to ICD10 conversion – that is, a mandated new standard of the International Classification of Diseases that is used for health insurance adjudication and reimbursement. As Diggs explains it: “This puts HIPAA and Y2K in the shade. We believe we can come up with a solution, and if we do, we can go to every insurer in the country.”

Other clients include Boeing’s PhantomWorks, Lockheed-Martin, European Media Laboratories (biotech research) and RZPD (part of the German equivalent of the US’s human genome project).

**Ontoprise: German engineering**

Ontoprise, based in Karlsruhe, Germany, is another university project turned commercial. The company was founded in 1999 by two professors and two PhD students from the university of Karlsruhe; two of them, Jürgen Angele and Hans-Peter Schnurr, are currently co-ceos. Its primary product is OntoBroker, an inference engine that serves as a semantic middleware and integration component within and between operational software systems. It uses F-logic, an optimized variant of frame logic. OntoBroker has more than 50 installations with a “more or less commercial background,” says vp sales & marketing Andreas Nierlich, although some are research projects. They account for about 80 percent of Ontoprise’s license revenues.
The company also offers OntoEdit, an ontology development tool that can read and write RDF (Resource Description Frameworks), frame logic, DAML+OIL, and database schemas. OntoEdit has more than 3500 users. That sounds like a high figure, but it includes the light version (no inferencing) that you can download free; a medium version ($590); and a professional version ($890), which does some light inferencing for testing an ontology.

“Ontology editors overall are becoming more of a commodity,” says Nierlich. Perhaps they are, in this leading-edge market. Whatever, OntoEdit creates demand for OntoBroker: It creates an ontology, but you need OntoBroker to integrate it into an IT environment to support industrial-strength applications with large data volumes.

Overall, the company has about 30 commercial customers, including Audi, Boeing, Bosch and Siemens and about 10 partners such as Absolute Software (Germany) who are using the technology to enhance their applications.

“It’s not our goal to model the world,” says Nierlich. “We and our related institute at the University of Karlsruhe have a lot of experience – research and commercial. We have come to the realization that we want to model very application-specific ontologies to do particular useful things.”

One customer willing to give a few details about using OntoBroker is Audi. Says Nierlich: “They have a system just now going into use for configuring prototype cars. There’s an engineer that wants to test a new car with a new engine, and needs a specific kind of car to fit that particular engine. For a given power, you need to match tires, brakes and gears. You need the right frame size for the engine to fit into the car. And you have to make sure this version of the software works with the control unit.” It doesn’t sound all that complex, but when you’re testing hundreds of engines a year, it helps to represent these requirements in software so that they can be determined and fulfilled quickly and accurately.

Another customer is Bosch, the well-known German producer of automotive technology. Bosch is using OntoEdit to model the outside world – i.e. the political and economic and social environment – and its likely impact on Bosch’s business for use in forecasting and planning.
Ontoprise has also sold OntoBroker to Absolute Software, a German software consultancy building avatars to act as travel agents. The idea is to combine dialogue with semantic technology and represent in an ontology more and more nuanced information than any single travel agent could ever have. “When you go on vacation,” says Nierlich, “you may not care if it’s Italy or Greece; you just want the sun and the sea. But most Websites ask you the location first.” The project is still in development, but it is already communicating effectively with a variety of online travel reservation systems. Eventually, it will include Web services to integrate realtime temperatures and other information.

Finally, Ontoprise has SemanticMiner, a query tool to improve information retrieval, to navigate the information and query expansion and so forth. It front-ends a variety of knowledge management applications and portals, and also comes as a plug-in for Microsoft Office, making it easy to integrate semantic information directly into the user’s work environment.

**What’s Next?**

All these tools are an illustration of the problem they are addressing: The world is not neatly organized, and local problems are best solved locally. Yet there are global problems, and we run into them more and more frequently in this increasingly interconnected world. Cyc’s approach, originally a centralized one, rapidly transformed into one of micro-contexts, where local rules and assumptions that might not be universal nonetheless hold true locally.

Just as the World Wide Web emerged from local efforts following only the barest of global standards, so will the Semantic Web and even the ontological Web emerge from local efforts. There won’t ever be global standards for many assumptions. But better communication of common assumptions and clear statements of the differences can still foster better understanding, whether human-to-human or program-to-program.

Over the next few years, ontologies are likely to remain a fairly esoteric field, but they are likely to show up frequently where the hard problems lurk – in health care, in design, in monitoring the real world. We need to be wary of what may be done in the name of security: With more and more devices capable of monitoring real-
world activities – from cargo and people movements, to financial transactions, purchasing behavior and e-mail or IM communications – technologists will not be able to shrink from facing tough ethical and social questions. With the power to model the world, we also gain some power to shape it. How should that power be used?
Release 1.0 Subscription Form

Complete this form and join the other industry executives who regularly rely on Release 1.0 to stay ahead of the headlines. Or if you wish, you can also subscribe online at www.release1-0.com.

Your annual Release 1.0 subscription costs $795 per year ($850 outside the US, Canada and Mexico), and includes both the print and electronic versions of 11 monthly issues; 25% off the cover price when you order from our online archives; a Release 1.0 binder; the bound transcript of this year’s PC Forum (a $300 value) and an invitation to next year’s PC Forum.

NAME _____________________________________________

TITLE ___________________________ COMPANY _____________________________

ADDRESS __________________________________________

CITY ___________________________ STATE _______ ZIP _________ COUNTRY ______________

TELEPHONE ___________________________ FAX _____________________________

EMAIL* ___________________________________________ URL ___________________________

*personal email address required for electronic access.

☐ My colleagues should read Release 1.0, too!

Send me information about multiple copy subscriptions and electronic site licenses.

☐ Check enclosed   ☐ Charge my (circle one): AMERICAN EXPRESS  MASTER CARD  VISA

CARD NUMBER ___________________________ EXPIRATION DATE ___________________________

NAME AND BILLING ADDRESS ____________________________________________

SIGNATURE ____________________________________________

Please fax this form to Natasha Felshman at 1 (212) 924-0240.

Payment must be included with this form. Your satisfaction is guaranteed or your money back.

If you wish to pay by check, please mail this form with payment to: EDventure Holdings, 104 Fifth Avenue, 20th Floor, New York, NY 10011, USA. If you have any questions, please call us at 1 (212) 924-8800; email us@edventure.com; www.edventure.com.