

Flexible decision support in dynamic interorganizational networks*

John Collins, Wolfgang Ketter[†], Maria Gini

Department of Computer Science and Engineering, University of Minnesota

[†]Department of Decision and Information Sciences, RSM Erasmus University

wketter@rsm.nl, [jcollins, gini]@cs.umn.edu

Abstract

An effective Decision Support System (DSS) should help its users improve decision-making in complex, information-rich, dynamic environments. We present a feature gap analysis of current decision support technologies, and we identify a set of DSS Desiderata, properties that can contribute both effectiveness and flexibility to users in such environments. We show that there is a gap between the features provided by current DSS technologies and the DSS Desiderata we aim for. We present a design-science approach that extends the boundaries of human decision-makers by creating a new and innovative artifact called “evaluator service networks” at the confluence of people, organizations, and technology. Our artifact enables users to compose decision behaviors from separate, configurable components, and allows dynamic construction of analysis and modeling tools from small, single-purpose evaluator services. The result is a network that can easily be configured to test hypotheses and analyze the impact of various choices for elements of decision processes. We have implemented and tested this design in an interactive version of the MinneTAC trading agent, an agent designed for the Trading Agent Competition for Supply Chain Management. We present an example of an evaluator service network that determines sales prices in a rich, dynamic trading environment. Additionally we

*A preliminary version of the paper (Collins et al., 2008b) was presented at the Smart Business Network conference in Beijing, May 2008, and won the best paper award.

describe visual interface elements that allow users to see and manipulate the configuration of the network, and to construct economic dashboards that can display the current and historical state of any node in the network.

Introduction

Organizations in business networks have a growing need for intelligent systems that can assist managers by gathering and analyzing information, making recommendations, and supporting business decisions. Advanced decision support systems and autonomous software agents promise to address this need by helping to compensate for human cognitive limitations and biases, resulting in better decisions. The recent advent of *Smart Business Networks* (SBN) (van Heck and Vervest, 2007; Vervest et al., 2008) extends the area of traditional business processes and gives rise to new challenges, especially in the area of dynamic and modular business process management, by enabling the integration of legacy systems and by providing advanced tools to facilitate human managerial decision making and inter-organizational integration.

An important element of the SBN vision requires flexible integration of technical infrastructure on an as-needed basis. However, it appears to us that the current generation of Decision Support Systems (DSS) is not up to the task in such a dynamic technical and business environment, for a variety of reasons. We are interested in finding approaches that provide the flexibility and robustness necessary to construct rich, effective decision support systems in the SBN world.

We present a gap analysis between the capabilities of currently available DSS technologies and the capabilities needed to meet future business needs. This analysis is based on literature review and discussions with experts in the field. Based on these results we have developed a flexible decision support technology called “evaluator service network.” With the advent of this technology we are able to make four major contributions to the SBN literature. First, a major element of the SBN vision is the ability of actors, by which we mean people, organizations, and their automation,

to quickly connect to other actors to achieve specific business objectives and then disconnect when a task is finished. Our first contribution in this paper extends the SBN literature through the design and practical implementation of a highly configurable and flexible decision support system that dynamically connects nodes of a business network and disconnects them when no longer needed. Our second contribution is an approach to goal directed service composition that takes advantage of a constrained service architecture that greatly simplifies the semantics of matching, composing and validating business services. Third, we are developing a tool that enables managers to visualize, understand, and validate service network compositions with a graphical representation of the actual network configuration. Fourth, we are developing a flexible economic dashboard architecture that can be dynamically connected to selected nodes in a service network to visualize their real-time status, such as current parts and finished goods inventory positions, risk and reward management, and the like. This architecture can greatly empower business managers in their understanding of the overall business network structure and facilitate real-time managerial decision making.

We have developed and demonstrated the effectiveness of evaluator service networks in a supply chain scenario. Traditionally, supply networks have been created and maintained through the interactions of human representatives of the various enterprises (component suppliers, manufactures, wholesalers/distributors, retailer and customers) involved. However, the recent advent of trading agents opens new possibilities for automating and coordinating the decision making processes among the various parties involved. Since experimenting on real world business networks is risky, and the real world is often a poor environment for controlled experiments, we have implemented and tested our evaluator service network on a supply-chain testbed, the Trading Agent Competition for Supply Chain Management (Collins et al., 2005) (TAC SCM). We describe an example implementation of our flexible decision support system and demonstrate its value using an interactive version of the MinneTAC trading agent (Collins et al., 2008a, 2009), an agent that performs coordinated buying, selling, production scheduling, and inventory management in the context of TAC

SCM. MinneTAC uses formal semantic descriptions of services that support service composition with verifiable outputs. We also present results of our network visualizer toolbox, where a manager is able to see and manipulate the current configuration of the network as well as the state of the different nodes.

In the following section we review the relevant related literature, focusing on the architectural challenges presented by interorganizational networks. We then present and discuss a feature gap analysis of currently available features in DSS, and our defined DSS desiderata. In the next section we present “evaluator service network” – our solution to this gap, evaluate the technology, and list multiple possible applications of it. We then highlight our contributions in different scenarios on an interactive version of the MinneTAC agent. Finally, we summarize our findings in the conclusions and present our research agenda for future work.

Related literature

This work draws from several fields. In computer science, it is related to software engineering, artificial intelligence, autonomous agents, and multi-agent systems, especially architecture, and the semantic web. In economics and information decision sciences, it draws from the framework of design science, smart business networks, and decision theory.

Design Science

Our research follows a design-oriented approach, as laid out by Hevner et al. (2004). With the design and implementation of the ontology-driven decision support system, we have created a valid artifact, which is relevant and necessary to solve existing problems in the information systems domain, because it has the potential to address each of the desired features identified in this section. Our presentation of the evaluator service network approach applies guidelines from (Hevner et al., 2004), including design as an artifact, problem relevance, and research contributions. Our

research contributes novel advances to the design of decision support systems, using a design-science approach.

Decision support systems

Much progress in technology and in business processes comes from automating the elements that do not require human judgement. Arnott and Pervan (2007) identify seven categories of DSS and, through extensive literature review, eight key issues in the DSS discipline.

Currently, the most prevalent way to interact with a computer is via direct manipulation (Shneiderman, 1983), a mode of operation in which the computer waits for specific commands from the user before performing responsive actions. Our vision is that we will be able to interact with a computer through dialog, much as we interact with people. This vision is based on agents that proactively collaborate with a user, in a mixed-initiative fashion, to predict the appropriate next steps that can speed up and improve the quality of a user's overall decision processes. Such agents are often referred to as interface agents (Maes, 1994; Shneiderman and Maes, 1997). We envision every user having readily-available personal Advocate Agents (Ketter et al., 2008) that are part of a supportive multiagent information system (Wellman et al., 1996). Such agents are semi-autonomous, reacting to additional input and feedback from the user.

Clark et al. (2007) discuss the importance of technology base and the gap between available and adopted technology for effectiveness of Management Support Systems (MSS), a framework that combines Decision Support, Business Intelligence, Executive Information Systems, and Knowledge Management Systems. We focus on the gap between currently available technology and the needs of advanced MSS.

Smart Business Networks

During the mid-nineties Goldman et al. (1995) and Sanchez (1995) stressed that in highly dynamic business networks the capability to quickly connect network actors (businesses) is essential to en-

able fast response times and greater variety when new opportunities arise. The concept of “quick connect” includes a search and select behavior by the different businesses. Goldman et al. (1995) further argue the need for a “quick disconnect” when the business transaction is over, otherwise open network connections can create undesirable information flows. At the time these articles were published no such network existed. Our architecture offers a unique way of automatically connecting, disconnecting and communicating with the appropriate actors in the network. One must pay particular attention to the interfaces of the different network actors. Establishing temporary connections between actors is infeasible unless interfaces match and the parties agree on the semantics of the data they share.

Kambil and Short (1994) argue that there is a strong need to construct software tools for business network representation, visualization, and analysis. These tools can help managers to visualize the different network actors, or roles, and linkage-based strategies of different organizations, enabling analysis of changes in emerging organizational forms. Hoogeweegen et al. (2006) and van Liere (2007) argue that knowledge of the network structure empowers the decision maker, and leads to better business decisions. Our architecture offers unique capabilities for network visualization, role and linkage analysis. Users can visualize the network structure, and drill down on particular actors to get a detailed picture of specific decision chains.

Creating performance and information dashboards (Eckerson, 2005) is part of the new emerging field of Business intelligence (BI) (Shmueli et al., 2006). BI provides functionalities such as real-time monitoring, performance reporting, and support for exploring solution space with normative models, statistical techniques and visualization. BI software can crawl the web, mine data, and generate reports customized to user preferences. According to Adam and Pomerol (2002) the layout of an economic dashboard has a direct impact on the understanding derived by managers. They argue that a graphical user interface (GUI) should be leveraged to maximize its visual impact. An extensive report on the visual design of dashboards has been presented by Few (2006). According

to Few many software companies, including Microsoft and Oracle, have sold dashboard applications since 2001 (see <http://www.enterprise-dashboard.com>). Our architecture fully supports BI and our dashboards are customizable for individual managers to facilitate managerial decision making.

Semantic web

According to Berners-Lee et al. (2001), the current hypertext document-centric web will evolve to include an infrastructure of machine-readable semantic descriptions that can be understood and acted upon by intelligent agents. In the last seven years we have seen significant progress toward the realization of this vision in the Semantic Web. We believe that conditions are now ripe for an explosion of useful Internet-based tools that go far beyond simple keyword search and human-driven information mashups. We foresee a day in the near future when sophisticated personal agents will run continuously in the background on our behalf, exploring, monitoring, filtering, mining, collaborating and presenting relevant information for our use, while flexible trust mechanisms will act to appropriately constrain the autonomous authority of those agents.

Service-oriented architecture has become an important technology for DSS, but there is debate among Information Systems professionals about its influence on managerial decision-making (Demirkan et al., 2008). Our work adds formal semantics to a service-oriented approach, dramatically changing its capabilities for flexible decision support.

Adomavicius and Tuzhilin (2002) present e-Butler, a first step toward a customer-centric architecture, but the architecture has not been implemented. These authors note that HTML is not fit for semantics, a limitation that is quickly disappearing with the advent of XHTML, micro-formats and presentation artifacts being moved into CSS-based display approaches. The authors promote XML, but they do not address RDF or other semantic technologies that we recommend in our proposed architecture. In contrast, our design leverages the power of network services, and encourages the creation of temporary business relationships through shared services.

Our work is complementary to that of Blau et al. (2009). Their approach addresses location, pricing, and provisioning of services in a public network. Our approach adds detailed, domain-oriented semantic descriptions of composable services and their dependency relationships. Huhn and Singh (2005) describe a process for combining heterogeneous services into solutions, and identify a number of problems with existing technology. Our approach addresses two of the problems identified in this work, by restricting the form of services and by incorporating formal semantic descriptions.

Universal Description, Discovery and Integration (UDDI) is a platform-independent, XML-based registry for businesses worldwide to list their Web-accessible services on the Internet. UDDI is an open industry initiative, enabling businesses to publish service listings and discover each other and define how the services or software applications interact over the Internet. Major drawbacks are the fact that the existing deployments are not for public consumption and the lack of rigorous semantic descriptions of services.

To facilitate the discovery and composition of services we need rigorous semantics based on a formal ontology. We think that usable general-purpose ontologies may be unrealistic, but that organization-specific and industry-specific ontologies are quite feasible and realistic. The Web Service Modeling Ontology (WSMO) (Roman et al., 2005; Vitvar et al., 2007) is a conceptual model for structuring semantic annotation of services and a step in realizing real world ontologies. The description of services at a semantic level using ontologies allows the use of machine reasoning and the use of implicit information in the process of partner selection.

Decision Support Systems: A feature gap analysis

Enterprises are economic entities run by humans. Humans have limited cognitive capacity and suffer from cognitive biases, resulting in limited ability to make rational or optimal choices (Simon, 1979). Instead of processing large amounts of information and reaching optimal solutions, humans tend to

use simple rules called heuristics to guide their decision making (Todd and Gigerenzer, 2001). As a consequence, humans are “satisficers” who typically are willing to accept a non-optimal solution that fits their needs, instead of “maximizers” who scrutinize and evaluate all the options available. Additionally, when humans are faced with a large number of options, they commonly suffer from the “tyranny of choice,” which leads people to experience negative emotions (Schwartz, 2004) and have difficulty in making choices. Findings conclude that while having some options is beneficial, too many options seriously impair the quality of the decisions made.

One should expect that modern Decision Support Systems (DSS) are designed to compensate these human limitations. In the following we highlight the most common technologies that are used for constructing DSS, and we then identify a set of “DSS Desiderata,” properties that affect the utility of DSS for their users. We are especially interested in DSS that can help their users to make rational choices in complex, information-rich, dynamic environments. In this sense we are primarily interested in Personal DSS, Negotiation Support Systems, and Intelligent DSS as defined by Arnott and Pervan (2007). Based on literature review, discussions with experts in the SBN community, and our own experience, we have identified a gap between the features provided by current DSS technologies, and the DSS desiderata we aim for.

Five technologies are commonly used for building DSS capabilities:

1. *Enterprise resource planning (ERP) Systems* are enterprise-wide computer software system used to manage and coordinate the resources, information, and functions of a business from shared data stores (Esteves and Pastor, 2001).
2. *Data Warehouse Systems* are subject-oriented, integrated, time-varying, non-volatile collections of data and query-based analysis tools, useful to support organizational decision making (Inmon, 2005).
3. *Spreadsheets* are desktop or web-based computer applications that model a rectangular grid of cells. Each cell contains alphanumeric text, a numeric value, or a formula that defines how

the contents of that cell is to be calculated from the contents of a combination of other cells whenever one of those cells is updated (Ragsdale, 2004).

4. *Expert Systems* attempt to reproduce the reasoning performance of one or more human experts, most commonly in a specific problem domain by capturing human expert knowledge in the form of rules and other formal structures, and providing an inference mechanism to generate consequences. This is a traditional application and/or subfield of artificial intelligence (Jackson, 1998).
5. *Mash-ups* are web applications that combine data or functionality from two or more sources into a single integrated application (Ketter et al., 2009; Mulholland et al., 2006).

All of these approaches have strengths and weaknesses. We analyze them in terms of a set of 12 specific capabilities or properties, our “DSS Desiderata,” shown in Table 1. We have categorized these capabilities as Software Engineering (separation of concerns, scalability, robustness, etc.), Smart Business Networks (van Heck and Vervest, 2007) (network structure visualization, ability to connect to and disconnect from services provided by partners, etc.), and Decision Science (transparency, ability to define and evaluate hypothetical scenarios). We have also numbered them, so that we can refer to them by number in the subsequent discussion.

In Table 2 we rate the identified technologies according to the desiderata we have identified. These ratings are admittedly subjective, based on “mainstream” applications of these technologies. There are clearly exceptions, and extremely technical users would disagree with some of our ratings, such as the perceived lack of abstractions and transparency in spreadsheet applications. Here we take the view of a non-technical mainstream business user.

There are clearly gaps in the feature set of these technological approaches, and some features are very poorly supported by any of these approaches. Some might argue that business users are not likely to make effective use of “programmer” features, such as user-defined abstractions and service composition, they do it with business processes and organizational structures as part of the

Area	Desired Property	Explanation
Software Engineering	1. Appropriate separation of concerns	Elements and tools are a good match to the skills and needs of both technical developers and non-technical business users.
	2. User configurability	System can be easily tailored to specific user preferences.
	3. User-defined abstractions	Users can factor out details so that one can focus on a few concepts at a time.
	4. Easy to experiment and test	Users can experiment with configurations and test the operation of the system at any time. Also called “exploratory development.”
	5. Robustness	System copes well with (unpredictable) variations in its operating environment with minimal damage, alteration or loss of functionality.
	6. Scalability	System handles growing workloads in a graceful manner, and/or can be readily enlarged.
Smart Business Networks	7. Network structure visualization	Users can visualize, understand, and validate the designed decision chain with a graphical representation of the actual network.
	8. Quick connect and disconnect	Users can dynamically connect and disconnect nodes of a business network according to current business needs.
	9. Goal directed service composition	Complex business applications can be semi-automatically composed and validated from individual services with formal semantic descriptions.
	10. Flexible dashboard architecture	User interaction tools can be dynamically connected to selected nodes to visualize and manipulate their real-time status.
Decision Sciences	11. Transparency	Users can see foundations and reasoning behind recommendations and decisions. The choice and presentation of information affects trust and confidence of human decision makers.
	12. Hypothetical scenarios	Users can substitute hypothetical for real-world data in order to predict impact of possible future events or courses of action.

Table 1: DSS areas: Software Engineering, Smart Business Networks, and Decision Sciences.

<i>Desired property</i>	<i>DSS Technology</i>				
	ERP System	Data Warehouse	Spreadsheet	Expert System	Mash-up
1. Appropriate separation of concerns	+	+	+	-	-
2. User configurability	0	0	++	+	++
3. User-defined abstractions	--	--	-	-	+
4. Easy to experiment and test	-	-	++	0	+
5. Robustness	+	0	-	0	-
6. Scalability	++	++	-	0	+
7. Structure visualization	-	0	-	+	-
8. Quick connect, disconnect	--	--	--	-	+
9. Service composition	--	--	--	0	+
10. Flexible dashboard	0	0	0	0	+
11. Transparency	-	-	-	+	-
12. Hypothetical scenarios	+	--	++	+	--

Table 2: Ratings of different DSS types and their desiderata. Rating system ranges from extremely difficult (--) through neutral (0) to easy (++).

management role.

There is another important element of decision support that is not effectively addressed by any of these technologies, with the exception of some expert systems. Most DSS are tools for human users - they gather, analyze, and organize information that is relevant for decisions human users must make. Autonomous agent technology (Bradshaw, 1997; North and Macal, 2007) can lead to systems that act on behalf of their users, adapting to their preferences, and interacting with other agents and network resources. Agents can be built with adjustable autonomy (Scerri et al., 2002), enabling users to build trust in the agent's recommendations. Agents can be designed to

monitor data in real-time, and take routine action without human intervention, and they can also act proactively, seeking to achieve the goals of their human users.

Filling the gap: Evaluator service networks

We have argued that existing DSS technologies are not sufficiently flexible, transparent, easy to work with, etc. to produce the kinds of ad hoc information and analysis that would truly leverage the skills and experience of their users, compensate for their cognitive limitations, and maximize their effectiveness. Our approach to closing the feature gap provides highly configurable, transparent decision processes that are fully described in terms that both end users and automated systems can understand. In our approach, information, analyses, and decision recommendations are composed from a large variety of data views and small, single-purpose analysis modules that can be composed into dataflow networks to produce results with well-defined business meaning.

We call these modules “evaluator services.” A schematic visualization of such a service is shown in Figure 1. Each evaluator service takes some input data from a variety of sources, performs some transformation on that data, and produces an output.

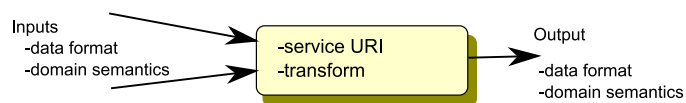


Figure 1: An evaluator service with its inputs, transform, and output.

Evaluator services (or just “evaluators”) can be composed into arbitrarily complex structures by connecting inputs to compatible outputs. Evaluators refer to each other by name rather than direct reference, and these names are configurable, either through XML configuration files, or through a user interface. This approach preserves independence among evaluator services, and it elevates and makes visible the high-level structure of the decision support processes. The result is that complex networks and feedback loops can be constructed from relatively simple services using metadata.

Figure 2 shows a simple example of such a network. The goal is to decide how much of a product to attempt to sell in the current market. The Sales model service recommends monthly sales goals, based on inputs including a prediction of future demand, a prediction of future prices, and the current and expected inventory of raw materials and finished goods. The Demand prediction service combines historical demand data with current economic projections to produce predictions of customer demand. This service could be developed in-house or purchased as needed. The Market price prediction service similarly combines input from multiple sources to predict market prices for the product.

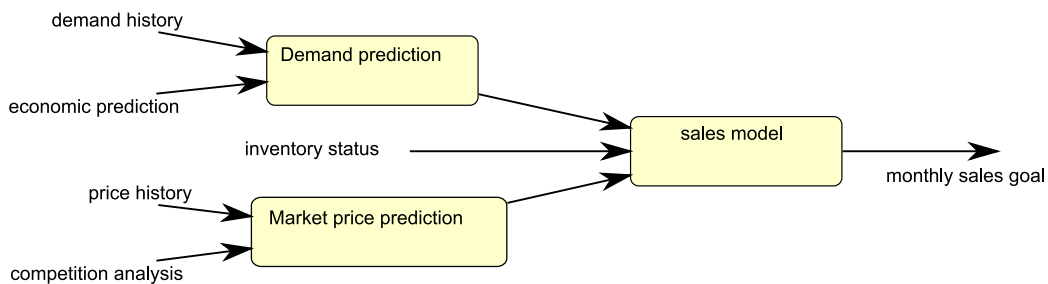


Figure 2: Simple example of an evaluator service network.

Service composition supported by semantic descriptions

Rich, complex decision-support systems can be built from simple, single-purpose analysis and modeling services, strung together in ad hoc dataflow networks. This basic capability addresses properties 8 and 9 (see Table 1) directly. The next step is to understand how this approach can be extended to construct a real-world decision-support system that can be useful in strategic, tactical, and operational business contexts. We envision seven basic capabilities for such a system:

- *Network composition.* Strategic and tactical users are able to add and remove services and their interconnections (property 2), and to define new abstract services by composing available services into sub-graphs (property 3), to support their own needs or those of subordinates. Network structure is visible and manipulable (properties 7, 11), and the simple structure

of evaluators makes it easy to experiment, and to incrementally build and test a complete system for a specific business purpose (property 4).

- *Control of individual services.* Strategic and tactical users can view and adjust parameters on individual services, such as interest rates or risk tolerance (property 2).
- *Dynamic composition of dashboard displays.* All types of users can add data viewers to individual dataflows. Drill-down capability would commonly amount to moving a viewer upstream in the network (property 10).
- *Direct injection of data.* Some managerial actions amount to deciding on the values of elements in a dataflow that represents a business process.
- *Alerts.* A notification service can monitor values within the service network, and send alerts to users or administrators that let them know when a metric falls outside predefined thresholds.
- *Hypothetical scenarios.* If a service network represents a business process with sufficient accuracy, strategic and tactical users can substitute simulated or historical data in place of real-world data on network inputs. This allows creation of hypothetical scenarios, allowing users to evaluate the likely outcomes of alternative decisions or anticipated market decisions, or to replay past events with alternate process models (property 12).
- *Intelligent agents with adjustable autonomy.* Evaluator service networks can be used to implement agents with communication and learning abilities, that can recommend actions or act on behalf of their users. Users need the ability to view and approve agent recommendations, and the agent must learn the preferences of its user in order to generate appropriate recommendations or actions.

We expect non-specialist users to be able to successfully build working evaluator service networks and understand the data that is produced. We also expect that software developers will implement

individual services. If the resulting services are easy for business users to understand and compose into effective decision and process support tools, this will provide effective separation of concerns between software professionals and business users (property 1). Business users who wish to compose solutions need to clearly understand the format, content, and business meaning of the data and of the transformations that are performed by the services. Much of the capability needed to express this information is provided by elements of the Semantic Web (Berners-Lee et al., 2001), specifically the Resource Description Framework (RDF), the Web Ontology Language (OWL), and associated inference tools (Yu, 2007). RDF represents simple facts (such as descriptions of services and data) as triples of the form [Subject Predicate Object] or alternatively [Resource Property-name Property-value]. For example, we could assert that a service `http://cs.umn.edu/svc/effdemand` provides a daily data element called `effective-demand` in the form of a vector of integers, indexed by product ID, that represents the current demand that could potentially be converted to sales at current prices:

Subject	Predicate	Object
<code>http://cs.umn.edu/svc/effdemand</code>	provides	<code>effective-demand</code>
<code>effective-demand</code>	update-frequency	daily
	datatype	<code>vector1</code>
	instance-of	vector
<code>vector1</code>	element-type	integer
	indexed-by	product-id
	represents	current available demand

and so on. The remainder of this description would detail the inputs and the transformation applied by the `effective-demand` service. A more detailed example is given in the next section.

Storing these facts in a uniform structure allows queries and logical inference to be performed on this dataset. As with web crawlers, an initial query can fan out and extract additional facts,

but there is one important difference. Unlike linked web pages, RDF statements require that each fact and link be a URI (e.g., the [provides] relationship is an abbreviation of the URI <http://cs.umn.edu/svc#provides> in the above example). By forcing precise and unique relationships to also be unique URIs, RDF queries can look for relationships between triples that have not been joined together. This simple and repeatable rule of triples can generate a complex, arbitrarily large “system of knowledge”. This semantic “web join” capability allows us to join otherwise unrelated data stored in general web page content.

RDF triples may be stored in structures called “triple stores” analogous to a single relational database table with three columns. Triple stores have their own query language called SPARQL (Query Language for RDF, <http://www.w3.org/TR/rdf-sparql-query>), which facilitates the creation of “mashups” of data. It does this by allowing discovery of nodes in separate semantic graphs that represent the same object. This in turn allows new facts to be generated from the RDF triples.

The evaluator service network design addresses most of the feature gaps we identified in the previous section. It can be implemented as a type of web-service mash-up, and shares the advantages of configurability, scalability, flexibility, and the ability to quickly connect and disconnect service nodes (properties 2, 6, 8). It is better at separation of concerns, because the simple, restricted form of evaluator services is much easier for non-technical users to understand and compose than the more general form of web services (property 1). Because of the simple network structure, it is easier to visualize and manipulate than a service mash-up (properties 2, 7), and the dataflow structure makes it much easier to construct hypothetical scenarios (property 12). Robustness is not directly addressed by this design, but the “blueprint” approach of Blau (Blau et al., 2009) can be used to improve robustness. In an inter-organizational environment, a security and location infrastructure will be required; we do not address the security implications in this work.

Semantic annotation of composable services supports key elements of the Smart Business Network vision. Emerging business needs can be addressed by semi-automatic composition of services

available from partners, vendors, and internal capabilities. Business users will be able to integrate these purpose-built networks into internal processes as needed, and will be able to control the type and degree of autonomy to optimize the balance between performance and risk. The automatic discovery, composition, and invocation of independent services will lead to an increase of the business agility of firms. We will expand on this approach using an example of service annotation and composition in the MinneTAC trading agent.

Building a trading agent with an evaluator service network

Since the inception of TAC SCM in 2002, more than 50 teams have built agents for the competition. These agents represent a variety of approaches to solving the various modeling and decision problems presented by the simulation scenario. MinneTAC (Collins et al., 2008a) models a flexible organization using the evaluator service network approach. There are a few top-level decision elements (Procurement, Production Scheduling, Sales) and a large number of evaluator services that act as modeling and analysis modules, supported by a common data store. For more details and additional examples we refer the readers to (Collins et al., 2009).

A high-level schematic representation of this design is shown in Figure 3. Decision components operate by retrieving data from the data store, and evaluation results from evaluators. The decision components themselves are implemented as evaluator services, and different implementations of them can be freely substituted into the network. Evaluators may request inputs from other evaluators, from the data store, and from external sources. They then transform that data in various ways, for example by updating price models, estimating demand trends, or composing and running optimizations to produce sales quotas or procurement recommendations. Results are provided in a common, self-describing format so they can be used by other evaluators or decision components. Connections among decision components and evaluators are entirely configurable and modifiable at runtime; the only real dependency in this design is on the data store, and on external data sources

such as market data and user inputs. This allows individual researchers to encapsulate modeling and decision problems within the bounds of individual evaluator services that have minimal, well-defined interactions among themselves.

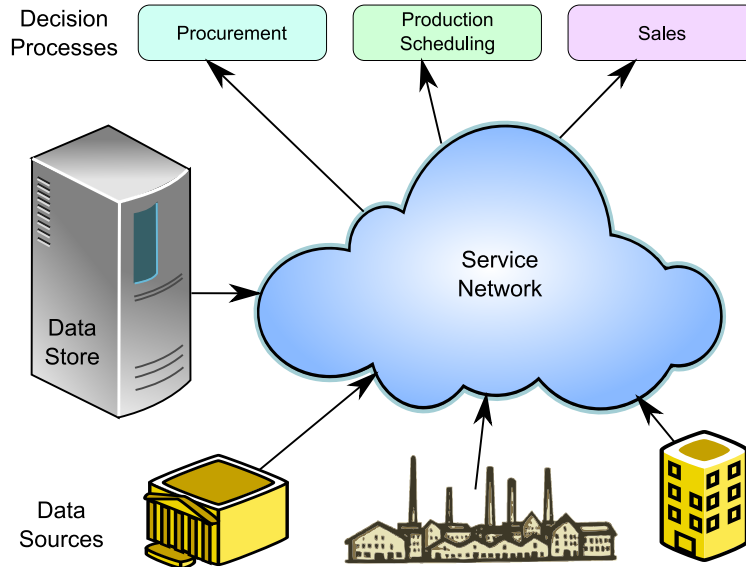


Figure 3: Abstract view of the MinneTAC trading agent architecture.

In Figure 3, the primary decision components are shown across the top. The Procurement service deals with suppliers, attempting to find the parts needed by Production Scheduling at the lowest possible cost. Manufacturing schedules the production facility with assembly tasks that maximize the expected value of its available inventory and production capacity. Sales sets prices and makes customer offers that are expected to maximize profit, given its available resources. These three decision services are in turn supported by a common data store and by a large set of interconnected evaluator services, represented schematically as the cloud in the center of the diagram. Specific examples of MinneTAC evaluator service networks are shown later in this paper. The evaluators, in turn, have access to each other and to various internal and external data sources, primarily in the form of periodic market reports that are issued by the simulation, and a data store containing a large body of historical data that has been “digested” by machine learning models,

such as the “economic regime” model described by Ketter (Ketter et al., 2007; ?).

Composing a MinneTAC evaluator service network

In this section we describe in some detail the annotation and composition of a small portion of the MinneTAC evaluator network, specifically the portion that determines sales offer prices. In a competitive auction market, an important decision variable is the probability that a customer will buy at a given price. Figure 4 shows schematically an instance of an evaluator that generates transfer functions called “pricers” that calculate a price for a given customer order probability. In other words, given a probability p , we can compute a price ϕ as $\phi = \text{pricer}(p)$. This is useful if one wishes to maximize profit by offering the highest price that can be expected to sell a target sales volume for a given level of demand.

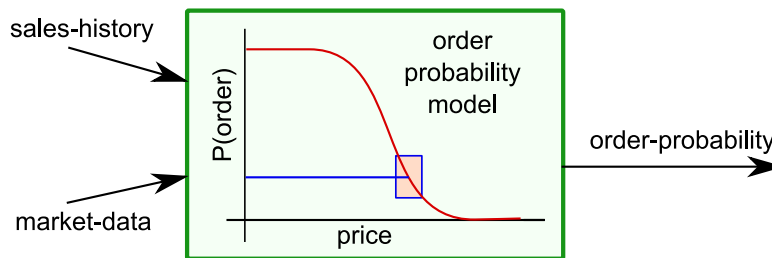


Figure 4: An evaluator service called order-probability that computes prices from an order probability model.

We can use such a transfer function to set an offer price for a product X . See Figure 5 for origin of variables.

1. Determine quantity Q_X (quotas) of product X to sell.
2. Determine the current demand D_X (current-demand) for product X .
3. Set price (product-prices) $\phi_X = \text{pricer}_X(Q_X/D_X)$ such that when offered on all demand D_X , moves Q_X units in expectation. This has the desirable side-effect of limiting prices available to competitors.

The evaluator that performs this function is called `simple-price` in Figures 7 and 8. Figure 5 is a snapshot of the portion of an evaluator network that computes prices in this way.

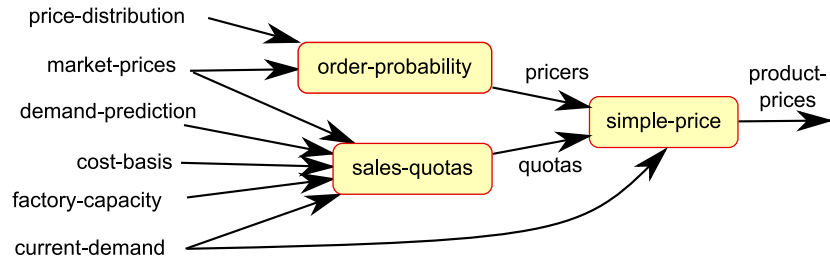


Figure 5: Evaluator example: Dynamic pricing chain.

In order to compose an evaluator network that includes the `simple-price` evaluator, we need to know what it expects for inputs, what it produces as output, and what function it performs. Knowing what it expects for input data, we can then locate other evaluators that produce that data and make the necessary connections. Figure 6 is a graphical depiction of the semantic description of the `simple-price` evaluator shown in Figure 5.

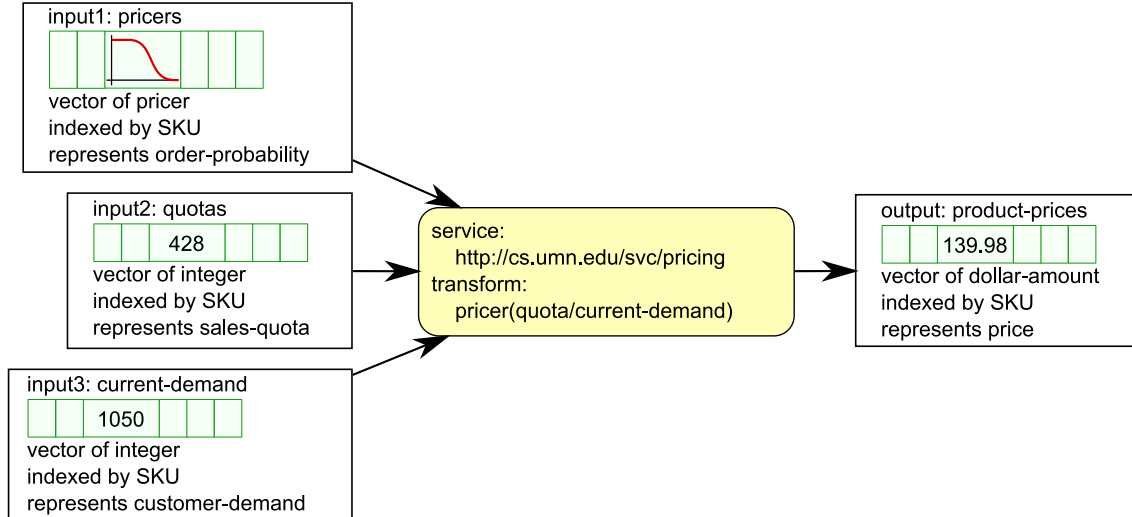


Figure 6: Semantic description of the `simple-price` evaluator.

In order to compose this evaluator into a network, it is necessary to find a match between each of the input specifications (`pricers`, `quotas`, and `current-demand`) and the output specifications of

candidate upstream evaluators. Of course, it will frequently be the case that exact matches cannot be found, especially in an open network. There are two basic types of inference that can be used to resolve mismatches:

1. *Semantic resolution.* The terms used in the description (SKU, sales-quota, customer-demand, etc.) may not match the terms used in the output descriptions of candidate services, but we may be able to discover assertions that will allow them to match. For example, if we are considering a demand measure from an entity called “Nielson,” we may have an assertion in our database that our term `customer-demand` means the same thing as Nielson’s term `expected-volume`.
2. *Format adapters.* If the mismatch is a datatype mismatch, we may be able to find a service that will transform the output datatype of a candidate service to our desired input datatype. For example, if we have a source of sales quotas that produces a 2-dimensional array of floating-point numbers representing sales quotas for the next 4 time periods, we can apply two successive transformations. The first would extract the quotas for the current time period, producing a vector of floats, and the second would convert the floating-point representation to integers.

Note that this relatively simple approach to service composition differs markedly from the more general methods described in the literature, such as (Traverso and Pistore, 2004; Rao and Su, 2004), because of the constrained dataflow structure of the services we propose.

To further illustrate the power of evaluator services, in Figure 7 we show a more detailed example of the evaluation network that is used to set prices in one of the MinneTAC configurations. Each of the cells in this diagram is an evaluator. Across the top of the diagram is a set of evaluators that estimate current market prices, future price trends, and the shape of the customer order-probability function, based on the method of “economic regimes” developed by Ketter (2007).

This configuration includes three different economic regime identification and prediction meth-

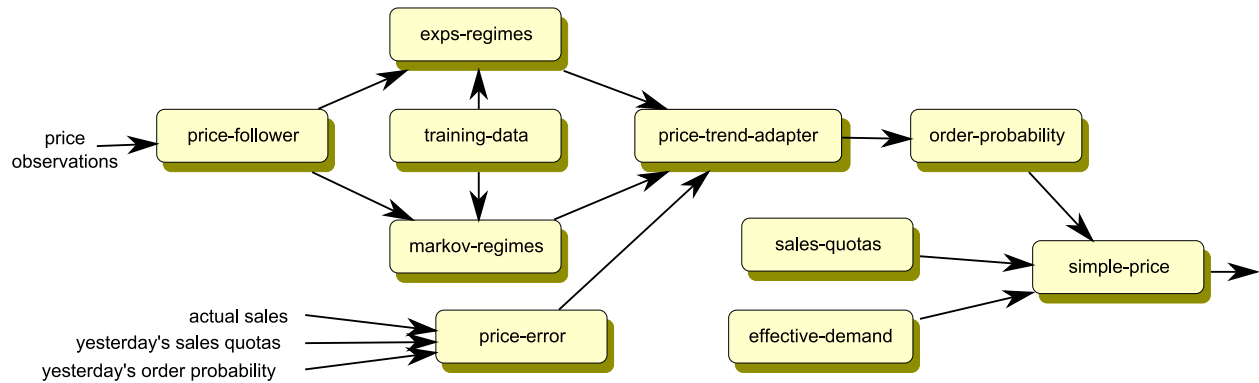


Figure 7: Evaluator chain for a sales manager that uses sales quota and information provided by regimes to determine prices, price trends and order probability.

ods. Two of them, the Markov prediction (MP) and Markov correction-prediction (MCP) models, are encapsulated in the `markov-regimes` evaluator. The third is an exponential smoother lookup (ExpS) process, encapsulated in the `exps-regimes` evaluator. Both of these evaluators depend on training data, distilled from a large number of past simulations. The `regime-training-data` evaluator supplies this data from an external data source. The analysis was developed using machine learning methods, as described by Ketter et al. (2007); ?. These evaluators can dynamically select the most appropriate portions of the training data for a given market situation. In a real business network setting we would train the system on historical transaction data, and update it at regular intervals. The economic regime method is able to identify current and predict future market conditions, such as scarcity and over-supply. Based on continuous regime probability predictions we are able to forecast current and future price distributions.

Visualizing evaluator service networks

As we can see from Figure 7, evaluator chains can become very complex. One result is that a design that was intended to make a complex agent design easy to understand has its own complexities. The original implementation requires users to configure an agent through a pair of XML files. One

maps implementations of the principal decision processes to their roles in the agent, and the other specifies the evaluators and their interconnections, along with parameters that control aspects of their operation. This is a major problem for two reasons. First, XML is not a particularly readable language for most people. Second, the network structure is not immediately evident from reading a configuration file. This leads to serious usability problems.

To ease the burden of creating and understanding these networks, we have built a visualizer and graphical editor for agent configuration files. Figure 8 is a screen shot of its user interface.¹ Users can visualize the entire network or portions of it, add or remove evaluators, and set parameters on the evaluators. The selected “price-error” evaluator uses the difference between sales quotas and actual sales to fine-tune sales prices.

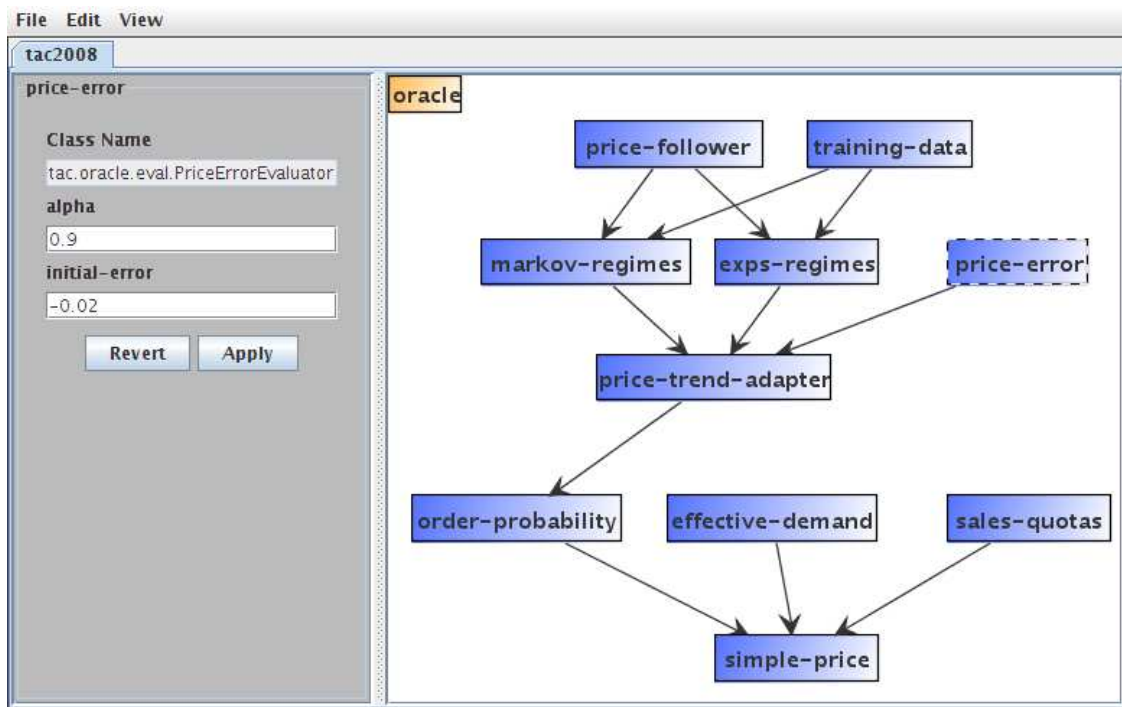


Figure 8: Network detail view of the agent configuration editor and visualization tool.

The ability to view and edit the evaluator network configuration is very helpful, but agent

¹Note that the configuration of a complete MinneTAC agent is much larger than this, typically including nearly 100 interconnected evaluator services.

designers, like business managers, need more than that in order to gain a clear understanding of the status of the business. Advanced business systems address this need by providing the ability to configure and display “dashboards” that can show summary and detail views of various quantifiable aspects of a business operation. The MinneTAC design allows a user to dynamically compose such “dashboard” displays by connecting a variety of graphing and plotting widgets to the outputs of evaluators. This can be done “on the fly”, while the system is running, because the composition of services (Sirin et al., 2003; Wu et al., 2003) and visualizations is entirely dynamic.

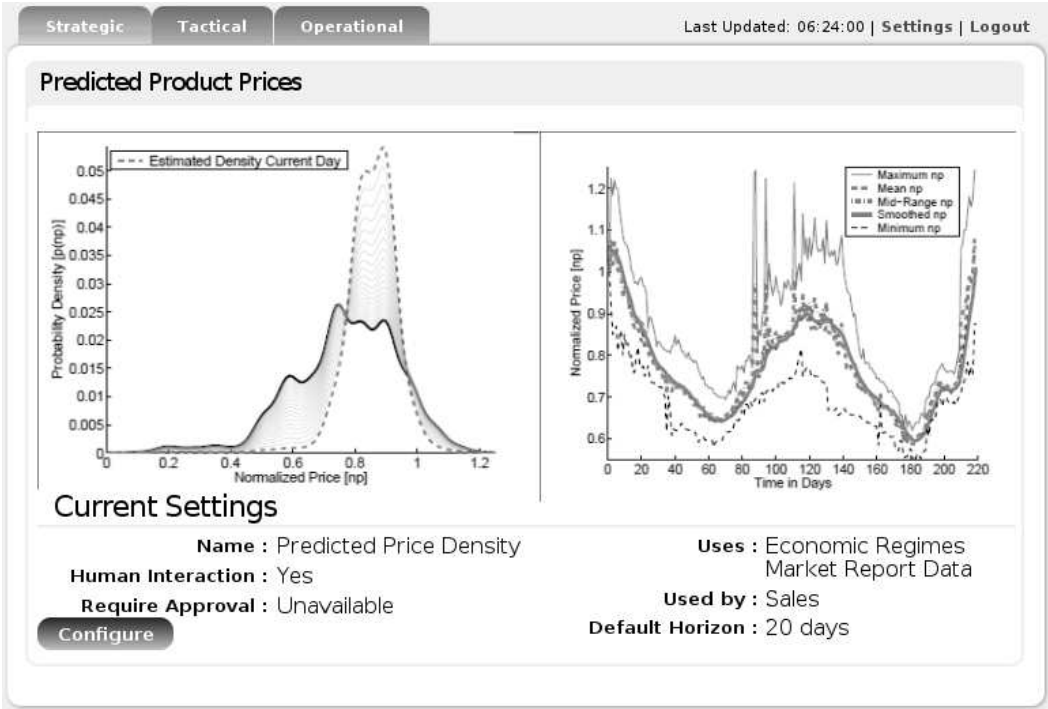


Figure 9: Dashboard showing current and predicted price distribution, and price history.

In an evaluator service network, virtually all of the quantitative information that drives decisions flows through connections between evaluators. This means that business-intelligence dashboards can be largely composed of relatively simple graphical viewers attached to these connections. For example, Figure 9 show a strategic dashboard with a sample price distribution prediction and historically observed prices that are used by the order-probability evaluator. In the left panel, the

dashed curve is the predicted price distribution for the current day, the solid thick line for 20 days in the future, and the thin lines for intermediate days. The shift of the price distribution towards the left over its 20-day horizon indicates decreasing future prices, and the increasing spread of the distribution reflects increasing uncertainty over time. The right panel shows a history of minimum and maximum prices for a specific product, along with three different statistical time series.

In addition to composing and viewing the decision network and its data, a business user will need the ability to view and possibly override the decisions it recommends. In Figure 10 we see a prototype interface that presents a user with recommended product prices for the current period, and gives the user the option of changing them.

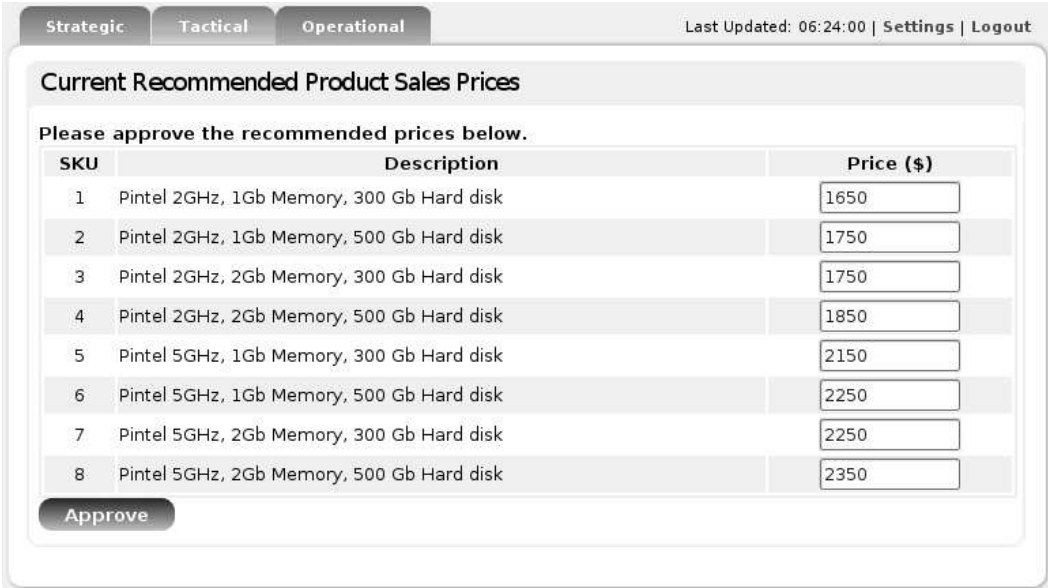


Figure 10: Interface for user approval of recommended prices.

Conclusions and future work

We believe that current approaches to building Decision Support Systems are not well-matched to the needs of a future environment where business agility will depend on the ability to build and manage business processes quickly, across organizations. We have identified a set of twelve “DSS

Desiderata” that will be important for building systems in organizations that wish to implement the Smart Business Network vision. We have shown that none of the current technologies for building DSS have all of these desirable properties.

We introduce an approach to closing this feature gap, in the form of evaluator service networks. This approach combines a restricted form of web services with a combination of technical and domain-oriented semantic description and inference. These evaluator services can be composed into dataflow networks to accomplish arbitrary monitoring, analysis, and decision support tasks ranging from simple data monitoring to fully-autonomous intelligent agents. Because an evaluator service network is composed of small, easy-to-understand components, and because the network itself is visible and manipulable, it is reasonable to expect that a non-technical business user will be able to understand, modify, and even create them to satisfy immediate business needs. Networks can be composed quickly from services drawn from the local technical environment, within an organization, or across organizations, and they can be easily disconnected when no longer needed. The combination of domain-oriented and technical semantic descriptions and inference tools will allow many kinds of problems to be solved by automatic or semi-automatic composition of evaluator networks. Because all the important data flows through identifiable network connections, it is easy to compose dashboards by connecting viewers to the outputs of individual evaluators. By routing a dataflow through a user interface element, recommendations can be presented to the user for approval with possible modification.

To demonstrate the effectiveness of the evaluator service network design approach, we have used it to implement a trading agent called MinneTAC that competes in the Trading Agent Competition for Supply Chain Management (TAC SCM). The flexibility afforded by the evaluator service network design allows MinneTAC to operate in either a mixed-initiative or fully-autonomous mode. The MinneTAC agent consists of an adapter that connects the agent to the simulation environment, a data store, and a large evaluator network. Most competent configurations of MinneTAC contain

between 60 and 100 evaluators. The TAC SCM environment is a severe test of software design and implementation, requiring hundreds of sales, procurement, and inventory-management decisions every 15 seconds. MinneTAC is competitive with the best agents that have been implemented for this scenario.

In complex economic scenarios such as TAC SCM, the desired design qualities include clean separation of infrastructure from decision processes, ease of implementation of multiple decision processes, clean separation of different decision processes from each other, and controllable generation of experimental data. In a competition environment, the ability to compose multiple agents with different combinations of decision process implementations makes it possible to quickly test hypotheses about the effectiveness of competing decision models.

The basic ideas behind the evaluator service network approach can be directly applied in a web service environment. The services could be implemented as RESTful services (Richardson and Ruby, 2007), supported by semantic descriptions expressed with OWL-DL (the description-logic version of the Web Ontology Language). Location and security services are required to complete the environment; their form and requirements will depend on the details of the application.

In the future, we plan to extend these ideas into an agent-assisted collaborative work environment in which a service network that supports business processes can be “tapped into” by multiple users to support their own roles and preferences. We also plan to support a rich capability for setting up and analyzing hypothetical scenarios with an array of simulation and statistical tools, implemented as evaluator services.

References

Frederic Adam and Jean-Charles Pomerol. Critical factors in the development of executive systems – leveraging the dashboard approach. In Manuel Mora, Guisseppi Forgionne, and Jatinder Gupta, editors, *Decision Making Support Systems: Achievements and Challenges for the New Decade*, pages

305–330. Idea Group Inc, 2002.

G. Adomavicius and A. Tuzhilin. An architecture of e-Butler: a consumer-centric online personalization system. *Int'l Journal of Computational Intelligence and Applications*, 2(3):313–327, 2002.

D. Arnott and G. Pervan. Eight key issues for the decision support systems discipline. *Decision Support Systems*, 44:657–672, 2007.

T. Berners-Lee, Hendler J., and Ora L. Semantic web. *Scientific American Magazine*, 2001.

Benjamin Blau, Clemens van Dinther, Tobias Conte, Yongchun Xu, and Christof Weinhardt. How to coordinate value generation in service networks? – a mechanism design approach. *Journal of Business and Information Systems Engineering (Wirtschaftsinformatik)*, Special Issue Internet of Services, in publication, 2009.

Jeffrey M. Bradshaw, editor. *Software Agents*. AAAI Press, 1997.

Thomas D. Clark, Mary C. Jones, and Curtis P. Armstrong. The dynamic structure of management support systems: theory development, research focus, and direction. *MIS Quarterly*, 31(3):579–615, 2007.

John Collins, Raghu Arunachalam, Norman Sadeh, Joakim Ericsson, Niclas Finne, and Sverker Janson. The supply chain management game for the 2006 trading agent competition. Technical Report CMU-ISRI-05-132, Carnegie Mellon University, Pittsburgh, PA, November 2005.

John Collins, Wolfgang Ketter, and Maria Gini. Architectures for Agents in TAC SCM. In *AAAI Spring Symposium on Architectures for Intelligent Theory-Based Agents*, pages 7–12, Stanford University, Palo Alto, California, March 2008a.

John Collins, Wolfgang Ketter, and Maria Gini. Flexible decision support in a dynamic business network. In Peter Vervest, Diederik van Liere, and Li Zheng, editors, *The Network Experience – New Value from Smart Business Networks*, pages 233–246. Springer Verlag, 2008b.

John Collins, Wolfgang Ketter, and Maria Gini. Flexible decision control in an autonomous trading agent. *Electronic Research Commerce and Applications*, 8(2):91–105, 2009.

Haluk Demirkan, Robert J. Kauffman, Jamshid A. Vayghan, Hans-Georg Fill, Dimitris Karagiannis, and Paul P. Maglio. Service-oriented technology and management: Perspectives on research and practice for the coming decade. *Electronic Commerce Research and Applications*, 7(4):356 – 376, 2008.

Wayne W. Eckerson. *Performance Dashboards: Measuring, Monitoring, and Managing Your Business*. Wiley, 2005.

J. Esteves and J. Pastor. Enterprise resource planning systems research: an annotated bibliography. *Communications of the Association for Information Systems*, 7(8):1–52, 2001.

S. Few. *Information Dashboard Design: The Effective Visual Communication of Data*. O’Reilly Media, Inc., 2006.

S.L. Goldman, R.N. Nagel, and K. Preiss. *Agile competitors and virtual organizations*. Van Nostrand Reinhold New York, 1995.

A.R. Hevner, S.T. March, J. Park, and S. Ram. Design science in information systems research. *Management Information Systems Quarterly*, 28(1):75–106, 2004.

M.R. Hoogeweegen, D.W. van Liere, P.H.M. Vervest, L.H. van der Meijden, and I. de Lepper. Strategizing for mass customization by playing the business networking game. *Decision Support Systems*, 42(3):1402–1412, 2006.

Michael N. Huhns and Munindar P. Singh. Service-oriented computing: Key concepts and principles. *IEEE Internet Computing*, pages 75–81, 2005.

W.H. Inmon. *Building the data warehouse*. Wiley, 2005.

Peter Jackson. *Introduction to Expert Systems (3rd Edition)*. Addison Wesley, 1998.

A. Kambil and J.E. Short. Electronic integration and business network redesign: a roles-linkage perspective. *Journal of Management Information Systems*, 10(4):59–83, 1994.

Wolfgang Ketter. *Identification and Prediction of Economic Regimes to Guide Decision Making in Multi-Agent Marketplaces*. PhD thesis, University of Minnesota, Twin-Cities, USA, January 2007.

Wolfgang Ketter, John Collins, Maria Gini, Alok Gupta, and Paul Schrater. A predictive empirical model for pricing and resource allocation decisions. In *Proc. of 9th Int'l Conf. on Electronic Commerce*, pages 449–458, Minneapolis, Minnesota, USA, August 2007.

Wolfgang Ketter, Arun Batchu, Gary Berosik, and Dan McCreary. A semantic web architecture for advocate agents to determine preferences and facilitate decision making. In *Proc. of 10th Int'l Conf. on Electronic Commerce*, pages 1–10, Innsbruck, Austria, August 2008.

Wolfgang Ketter, Marko Banjanin, Rob Guikers, and Alfred Kayser. Introducing an agile method for enterprise mash-up development. In *Proc. of the IEEE Conf. on Commerce and Enterprise Computing*, Vienna, Austria, July 2009.

P. Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30–40, 1994.

A. Mulholland, C.S. Thomas, P. Kurchina, and D. Woods. *Mashup Corporations: The End of Business as Usual*. Evolved Technologist Press, 2006.

Michael J. North and Charles M. Macal. *Managing business complexity: discovering strategic solutions with agent-based modeling and simulation*. Oxford University Press, USA, 2007.

C.T. Ragsdale. *Spreadsheet modeling and decision analysis*. Thomson/South-Western Mason, Ohio, 2004.

J. Rao and X. Su. A survey of automated web service composition methods. In *Proc. of the First Int'l Workshop on Semantic Web Services and Web Process Composition*. Springer, 2004.

Leonard Richardson and Sam Ruby. *Restful web services*. O'Reilly, 2007.

Dumitru Roman, Uwe Keller, Holger Lausen, Jos de Bruijn, Rubn Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Cristoph Bussler, and Dieter Fensel. Web service modeling ontology. *Applied Ontology*, 1(1):77–106, 2005.

R. Sanchez. Strategic flexibility in product competition. *Strategic Management Journal*, 16:135–159, 1995.

Paul Scerri, David Pynadath, and Milind Tambe. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17:171–228, 2002.

B Schwartz. The tyranny of choice. *Scientific American*, 290(4):70–75, 2004.

G. Shmueli, N.R. Patel, and P.C. Bruce. *Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner*. Wiley-Interscience, 2006.

Ben Shneiderman. Direct manipulation: A step beyond programming languages. *IEEE Computer*, 16(8):57–69, 1983.

Ben Shneiderman and Pattie Maes. Direct manipulation vs. interface agents. *interactions*, 4(6):42–61, 1997.

Herbert A. Simon. Rational decision making in business organizations. *The American Economic Review*, 69(4):493–513, 1979.

Evren Sirin, James Hendler, and Bijan Parsia. Semi-automatic composition of web services using semantic descriptions. In *Web Services: Modeling, Architecture and Infrastructure at ICEIS*, 2003.

- P.M. Todd and G. Gigerenzer. Précis of simple heuristics that make us smart. *Behavioral and Brain Sciences*, 23(05):727–741, 2001.
- P. Traverso and M. Pistore. Automated composition of semantic web services into executable processes. *Lecture Notes in Computer Science*, pages 380–394, 2004.
- Eric van Heck and Peter Vervest. Smart business networks: how the network wins. *Commun. ACM*, 50(6):28–37, 2007.
- Diederik van Liere. Network horizon and the dynamics of network positions: A multi-method multi-level longitudinal study of interfirm networks. Technical report, RSM Erasmus University, ERIM, 2007.
- Peter Vervest, Diederik van Liere, and Li Zheng. *The Network Experience – New Value from Smart Business Networks*. Springer Verlag, Berlin, Germany, 2008.
- T. Vitvar, A. Mocan, M. Kerrigan, M. Zaremba, M. Zaremba, M. Moran, E. Cimpian, T. Haselwanger, and D. Fensel. Semantically-enabled service oriented architecture: concepts, technology and application. *Service Oriented Computing and Applications*, 1(2):129–154, 2007.
- Michael P. Wellman, Edmund H. Durfee, and William P. Birmingham. The digital library as a community of information agents. *IEEE Expert*, 11(3):10–11, 1996.
- D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating daml-s web services composition using shop2. *Proceedings of 2nd International Semantic Web Conference (ISWC2003)*, 2003.
- Liyang Yu. *Semantic Web and Semantic Web Services*. Chapman&Hall/CRC, 2007.