A Market Mechanism for Participatory Global Query: A First Step of Enterprise Resources Self-Allocation

Cheng Hsu*, Christopher D. Carothers** and David M. Levermore*

*Decision Sciences and Engineering Systems, hsuc@rpi.edu and leverd@rpi.edu
**Computer Science, chrisc@cs.rpi.edu
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

Information Technology and Management, 2004

Abstract

The problem of Database Query has always been considered from the user's side. That is, the databases are always treated merely as the object of search, rather than being a subject or willing participants of an information exchange. This paradigm works when all participating databases belong to a single authority (such as a company) under which their participation is definitive and their contents completely open for the querying. Traditional single databases, federated databases, and even the new XML-based Internet databases subscribe to this user-oriented paradigm. However, emerging information enterprises are increasingly ***collaborative*** in nature, since they tend to involve, on a real-time and on-demand basis, a large number of databases belonging to many different organizations whose participation is conditional and case-by-case; e.g., drilling through supply chains. These collaborative queries deserve a new paradigm that equally account for the provider side. Research has shown that market-style self-allocation of users to providers is a promising approach to support such a paradigm. However, previous results of artificial markets are insufficient for global database query. Therefore, we develop an artificial market model to provide a ***Two-Stage Collaboration*** solution, where the first stage establishes optimal participation of databases for a search task, and the second executes the task in a traditional database query manner.

The proposed model employs a new agent-based, peer-to-peer publish and subscribe approach to self-allocating database resources in an information enterprise. This approach promises to lead eventually to allocating other classes of information resources, as well. New results include (1) an agent model using a Metadatabase and an Agent-Base to create and manage large number of custom agents, (2) a peer-to-peer negotiation method, and (3) an open common schema design. The paper also provides an implementation scheme for developing the artificial market. Laboratory tests show that such a mechanism is feasible for large scale matching and negotiation as required by the first stage. The second stage employs mainly previous results established in the field.

Key words: market-style scheduling, participatory database query, agent-base, peer-to-peer market, common schema, publish-and-subscribe model, Metadatabase

1. Provider Participation in Database Query

Market economics represents a new thinking in the struggle for real-time scheduling. Some pioneering efforts are described in works for computing and manufacturing [Clearwater 1996]. A common theme in these results is to turn the traditional paradigm of algorithms upside down: Instead of starting with the users (jobs) on the top and pushing their needs down to providers (resources) and thereby centrally commanding the resources to satisfy the needs of the jobs, the market-style solutions call for the jobs to bid for the resources under the latter's purview. In this sense, we might conveniently refer to the traditional paradigm as a command economics. The value for such a market style – or, provider participation – paradigm is often analyzed in these works to be the improved performance. The mechanism by which such improvements are achieved includes the simplification of the command chain, which reduces built-in waiting time, and the dynamism of the control criteria, allowing for flexible utilization of resources. We would further argue that for large-scale enterprise information exchange, a market mechanism has the potential to promote proactive participation and hence maximum utilization of all available information resources in the enterprise, regardless of their being a part of the formal "information shops" or not, through its pricing or rewarding models.

The term information shop here is used to refer to the usual configuration of databases for information integration in an enterprise; such as a database server and its clients, or a formally structured multiple databases system. Just like machines in a work shop, the databases of an information shop participate in the shop completely, persistently, and unconditionally. However, an enterprise usually has many other databases outside of an information shop that also possess valuable information resources for tasks done in the shop. It is practically intractable for any large scale enterprise to formally include all relevant databases in pertinent information shops all the time. It becomes virtually impossible when the enterprise extends itself to interoperate with its constituencies on an online and real time basis. The national security community provides an example. The community includes some major agencies such as the NSA, FBI, and CIA. But it could involve other agencies at the federal, state, and local levels such as police departments. Many unexpected, non-designated sources might also contribute when certain situations arise. Therefore, even when all 34 or so national security databases of these major agencies are integrated in some manner, there will always still be needs to engage other information sources on an on-demand basis. The market approach is poised to encourage all these "unusual suspects" to play and gain, as well as mobilizing the usual ones.

Traditional database query, be it for single databases, multiple databases of an enterprise, or Internet databases, follows the command economics paradigm in the above sense. It works well for information enterprises that subscribe to its premises and satisfy its requirements. We explore the problems that do not fit so well with the traditional paradigm; in particular, we consider the information enterprises where virtual information shops – ad hoc and task-based involvement of databases – represent a major need, as in the national security community, global news media organizations, and industrial exchanges such as Covisint. Incidentally, no industrial exchange today is capable of global query of databases. The database-focused problem for the paper is,

therefore, ***the global query of a loose collection of databases whose participation in the information exchange and sharing is controlled by the databases***; or the **participatory database query problem.** We do not impose conditions on the number of databases and the nature of their networking, nor the regime of data integration such as federated schemas, data dictionary, or XQuery in this definition. These important issues belong to the design of the specific solution algorithms for particular application domains and requirements; and they define the particularization of a general solution for this class of problem. We propose the general solution, called **Two-Stage Collaboration**, below:

Define: **Enterprise Resources Market** is a set ERM(A,M,B,S,P), where
A: a structure and algorithm of software Agent representing an information requesting task and/or information provision task at an originating site (user and/or database), and being executable by the Blackboard and by the proxy Server at the database;
M: a structure and algorithm of enterprise metadata, or a Metadatabase, sufficient for integrating data semantics at the performance level required by the applications and for interoperating with Agents;
B: a structure and algorithm of market administration, called the Blackboard, capable of executing the first stage and the second stage solutions defined below;
S: a structure and algorithm operating at the local site as an added shell to the database, called a proxy Server, responsible for interoperating the Blackboard with the database and managing the agents and metadata for the database; and
P: a structure and algorithm, called the Price and Performance model by which the Market valuates the resources and information exchange and sharing, and measure the overall performance of the system.

The basic logic of the Two-Stage Collaboration solution using ERM:
The *objective function*: the maximization of the total (perceived) value of information and physical resources.
The *constraint*s: the transaction requirements and data semantics of tasks.
A *feasible solution*: a match of databases (providers) with information requests (users)
The *optimization*: Execute for a request for information (demand bid) or a provision of information (supply bid),
Stage 1: find the optimal involvement of local sites (users and/or databases) by matching and negotiating the demand and supply bids carried by agents at the Market.
Stage 2: determine the suitable regime of database query for the scope of database participation defined in Stage 1 and process the query accordingly.

The Market's ability to execute concurrent matches and queries, either in a peer-to-peer manner or in global, is a function of the specific design and its implementation. We might comment that a solution for Stage 1 has the chance to also incorporate the best e-auction results from the field (e.g., [Sandholm 2003]) to fit the particular applications. Stage 2, on the other hand, corresponds closely to the traditional database query problem and can be processed in the traditional paradigm, including some of the newer results (e.g., [Stonebraker, et.al 1996 and Braumandl et.al. 2001]). Extensions are, of course, required to integrate them with the first stage; such as assure closure of tasks when concurrent jobs or peer-to-peer negotiation are involved, among other things. We develop a particular

design in this paper as a first solution to the participatory database query problem and hence a contribution to the field.

In a broader sense, databases are just one class of enterprise information resources; but they are a core class. Other classes include computing resources, networks, and even human experts (or the knowledge they possess and express in digital means such as emails). We switch to this broad view to analyze the problem of self-scheduling for information enterprises and discuss the related literature in the next section, Section 2. Section 3 describes how the general solution works when implemented. A detailed design of the Market aiming to make provider participation work for multiple databases in an Internet-based extended enterprise is presented in Section 4. Whenever appropriate, the discussion also points out how the design can handle the broad resource allocation problem. A key challenge of the design – large scale concurrent matching – is tested in laboratory through a prototype match engine and the results are described in Section 5. Along with the analysis for implementation that Section 4 also provides, the results of these two sections, 4 and 5, represent an evidence for the soundness of the proposed ERM model and the Two-Stage Collaboration solution. The last section, Section 6, discusses the further development of the solution, its application to Internet Databases and single databases, and its possible extension to allocate other enterprise resources.

2. The Problem of Market-Style Self-Scheduling

To understand market-style participatory database query we need to first understand market-style resource allocation in general. For the purpose of this paper, information enterprises are exemplified in the intelligence community, news organizations, the ASP (application service provider) model of e-business [Tao 2001], industrial exchanges (e.g., Covisint, FreeMarkets, and CommerceOne) [Glushko et.al. 1999], and the service business of heavy manufacturers (e.g., Boeing, GE Industrial Systems, and Samsung Electronics) [Hsu and Pant 2000]. At the heart of these enterprises are their information resources, which may require open and scalable regimes to inter-operate them. These regimes, including enterprise-wide scheduling and control, allow an information enterprise to operate at high level of integration and to capture the full benefits of extended enterprising. The problem of resource allocation for information enterprises exhibits the unique nature of information production, different from those for the production of physical goods such as manufacturing. We can characterize the problem as resource allocation under the conditions of globally distributed dual-nature participants (resources and users), heterogeneous information models, and real-time assignment with online performance measurement and adjustment – we refer to it as the **enterprise resource allocation problem** [Hsu and Carothers 2003]. The participatory database query problem defined in Section 1 is a core subset of this problem. Since global query per se is well understood in the literature (see, e.g., [Stonebraker et.al. 1996, and Cheung and Hsu 1996]), we analyze the participatory problem in the light of this general problem.

The enterprise resource allocation problem defies many premises of classical scheduling theory [Conway et.al. 1967] and online scheduling [Hochbaum and Shmoys 1987, and Coppersmith and Raghavan 1987]. The classical paradigm focuses on optimizing the

supply (resources) with respect to a given workshop-scale ***batch*** demand (tasks), subject to workflow precedence and other job constraints. This formulation leads to a few basic assumptions that do not hold for information enterprises. First is the dichotomy between resource and user where, as in the case of manufacturing, machines and jobs are two orthogonal genres and a job could not be a resource and a resource not a job. Second, the paradigm does not consider individual instances of each genre, but only the characteristics of classes of jobs or resources, such as machining capacity, processing times, and due dates. Finally, a traditional scheduler is a planner that does not have to consider real-time conditions nor online feedback from the system (shop floor) that executes the schedules. The assumption is that either the system's controller will adjust the schedules to accommodate real time conditions or the scheduler will re-run itself with new conditions to produce a new result in the next planning window. Under these assumptions, matching a job to a machine is never a problem, and the problem is but how to optimize the allocation (batches of jobs to batches of machines) in terms of throughput, make-span, tardiness, utilization rate, and other performance measures. When necessary, such as in online scheduling of jobs for computers, an assignment can be moved around as in bin packing within a single migration round, because homogeneity is a given.

Enterprise information resources do not fit with any of the basic assumptions of the conventional scheduling paradigm. A user of an information enterprise could also be a resource (provider) for it and vice versa. For example, a field officer could both provide and request information and an automated analysis system or a database might request input or even co-processing with other facilities as well as produce output. Next, information resources could be highly heterogeneous and individual differences count; e.g., these providing or requesting tasks might use different data semantics to describe their information contents and requirements. Furthermore, the scheduling and execution cannot be separated but have to be connected with synergistic information sharing and flow. In addition, information enterprises could involve very large numbers of participants especially when extended organizations and globally distributed resources and users are involved. Under the above predicates, the scheduling regime must produce maximum quantity of information to maximum suitable users with best quality and least delay. The regime must also encourage information sharing among individual entities as well as support pervasive connection of participants to achieve synergistic scheduling and execution. In short, it must ***dynamically balance*** the supply and the demand. We submit that these characteristics fit best with an artificial market model, whose instances include stock exchanges or industrial exchanges in e-business.

In fact, a number of researchers have recently proposed market-style resource allocation schemes using software agents to make manufacturing scheduling models more in line with large-scale, real-time assignment [Baker, 1996, 1998; Swaminathan et.al., 1998; Cesta et.al., 2000; Prabhu 2000; Nandula and S. P. Dutta, 2000; Smith et.al. 2000; Parunak, 2001; Heragu, et.al. 2002]. More authors have focused on computing, or simply the general paradigm of allocating resources through an artificial market (e.g., [Walsh and Wellman 1998] and papers in [Clearwater 1996]). These newer efforts tend to create a pseudo market for, e.g., shop floor scheduling and networked computing resource allocation where facility agents and job agents meet and match. Many results either stay

at a generic or conceptual level without attempting a feasible design for a practical problem or do not provide sufficient results (e.g., a performance-feedback-reward loop or data semantics resolution) necessary for approaching global optimality - i.e., equilibrium not only in prices but also in operation of resources. Researchers have noted that some of these pseudo markets exhibit various global inefficiencies (e.g., long queues at certain resource sites caused by obsolete information at the global site, and tasks not getting assigned properly due to lack of negotiation); and others incur excessive overhead (imposition of a global controller to supplement self-scheduling). An effective market mechanism for ***information sharing*** is still evasive in the filed. At the design level, some of these shortcomings in some of the previous results could be attributed to their agent models; which are insufficiently adaptable for performing dynamic operations required by information enterprise resource allocation; e.g., semantics matching and task constraints updating. Another requirement that is also not met in the previous literature is the capability to create, update, and process very large number of software agents in a concurrent manner. This requirement is due to the large number of participants who might take part in the market on an ad hoc and on-demand basis. We now turn to discuss the general structure of the ERM model.

3. The General ERM Model: Agent-Based, Peer-to-Peer Publish and Subscribe

The setting of the general ERM model is summarized this way. It considers an information enterprise as encompassing multiple operating groups, databases and computing networks; all of whom have information contents and can process data. All participants can be both providers and requesters of information resources. The organizations use a reward system to control the allocation and utilization of information resources. All users pay from their funds (real money or fungible credit) for their requesting tasks (buy) and take revenue from their offering tasks (sell). The organizations could make online adjustments of funding for participants, as desired.

Participants use task-oriented and manageable software agents to publish their requests and offers of resources, and use the same to subscribe to the resources. Software agents bring about several significant advantages: asynchronous (24/7) transaction, controllable consistency with enterprise knowledge and requirements, instantaneous response time, and security (e.g., the participant can publish only the information slated to offer, and conceal the true nature of the tasks or identity of the requesters from the providers if necessary). The design of the agents must also include additional intelligence such as preferences and transaction rules to further automate negotiation and peer-to-peer transaction as described below.

The model calls for a global server, or the market/exchange site, where a blackboard is maintained to execute the two-stage collaboration solution (see Section 1). For the first stage, a key algorithm at the blackboard is the matching of requests and offers based on task characteristics such as the representation of the task, deadline, specific requirements, availability, and perceived value in terms of price. After one or more matches are found, a pricing model will optimize the assignment through a negotiation algorithm, which can include group auction and revision of terms among the matched parties if alternative

allocations exist. The market maintains and makes available the market status to all participants to help them publish their bids and negotiate, as well as provide remedy to tasks having difficulty getting allocated. For instance, the server could make an automatic "loan" to add to the price offered by a request to find it a match just before the deadline. The loan becomes a feedback to the reward system on the initiator of the task. After finalization of the assignment, the processing enters the second stage and the agents proceed to establish connections for the requesters and the providers' resources at the local level. A proxy server of adjustable complexity could reside at the local resources to enable peer-to-peer transactions. The requests become jobs at the local resources and queue themselves according to the price offered and follow the local queuing discipline. The processing at the local sites does not require global coordination. For database query, the requests are global queries. Therefore, the blackboard at the second stage executes a global query algorithm over the participating local databases. The algorithm will translate the original requests into local database queries and assign the latter as jobs to the appropriate proxy servers. The queuing status, including workload, will become feedback to update the local resource's agents at the blackboard. Proxy servers will also return the results either to the blackboard or to peers, directly. The market also contains a Metadatabase storing enterprise metadata about tasks characteristics and enterprise requirements to support the blackboard, and an Agent-Base to create and manage the task agents online according to users' instructions. The global server then execute the global query algorithms for the second stage solution – see Section 4 for details.

The model allows for an alternative to going through the Black Board; i.e., a participant can initiate directly task agents that visit other task agents at other local sites to find-negotiate the matches and conduct auction when necessary, on their own within the virtual "match circle" (a virtual information shop). This alternative is available to local sites that have sufficient computing power. In this case, the initiating participants will control the virtual auctions that their task agents started first (distributed computing); and hence simplify the computing load at the global server.

The above is an overview of the operation using the ERM model. The basic architecture of the ERM is shown in Figure 1. The market comprises its own operating domain, the Exchange Domain, and has full control of this domain. The enterprise participants control their own local operating domains, the Participant Domain, which is delegated to them by their respective organizations. The organizations control fully their own operating regimes, the Enterprise (Agency) Domain. As such, the Exchange Domain only inter-operates with the Participant Domain as mutually agreed upon but does not otherwise intrude it. Firewalls could exist between these domains. Similarly, the Exchange Domain could support the Enterprise Domain according to some charters; otherwise, they are separated from each other. The architecture includes Internet as a means of networking between the participants and the artificial market only for the sake of generality. Other means owned by the Agency could replace (or augment) the Internet without altering the overall design of the model. Figure 2 and Figure 3 show, respectively, the basic structure of the participant side of the architecture and the exchange side of it.

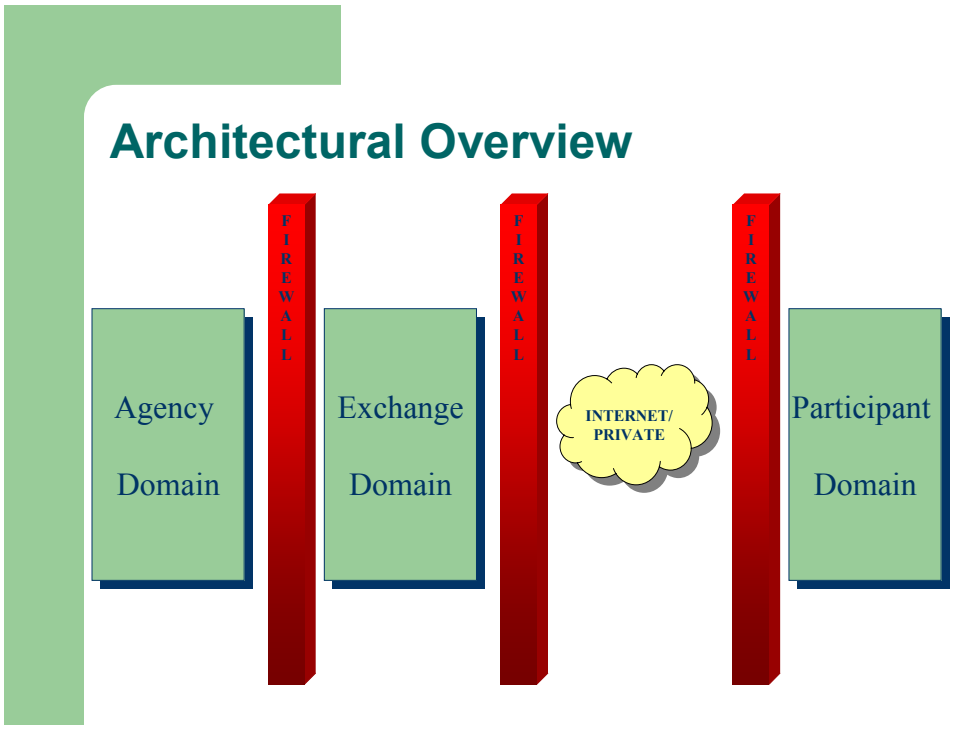

Figure 1, the Separate Domains of the Enterprise Community.

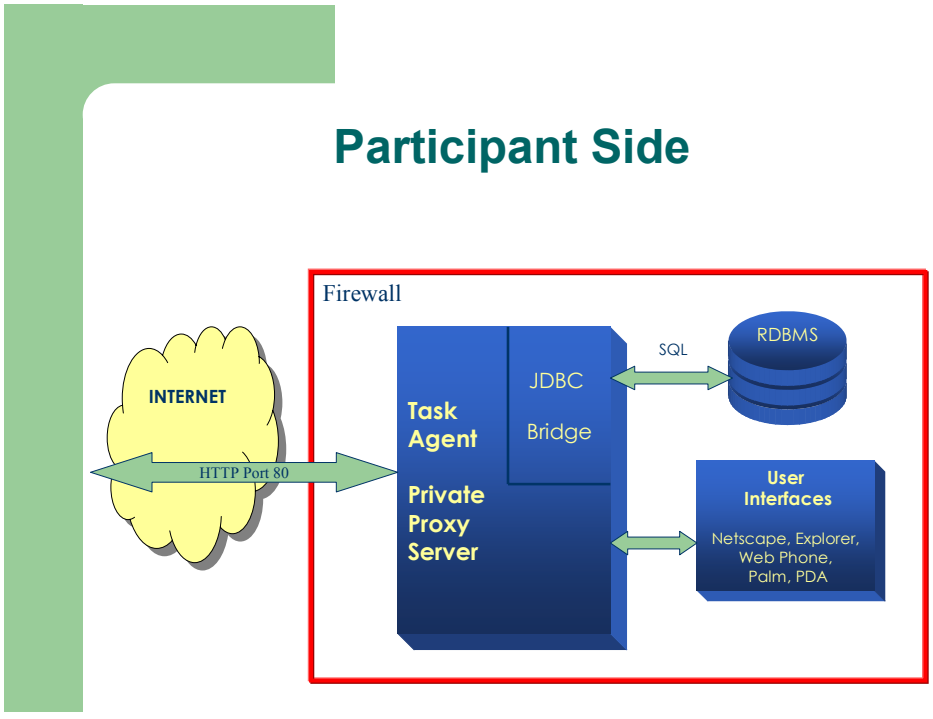

Figure 2, the Interaction with the Participant Side.

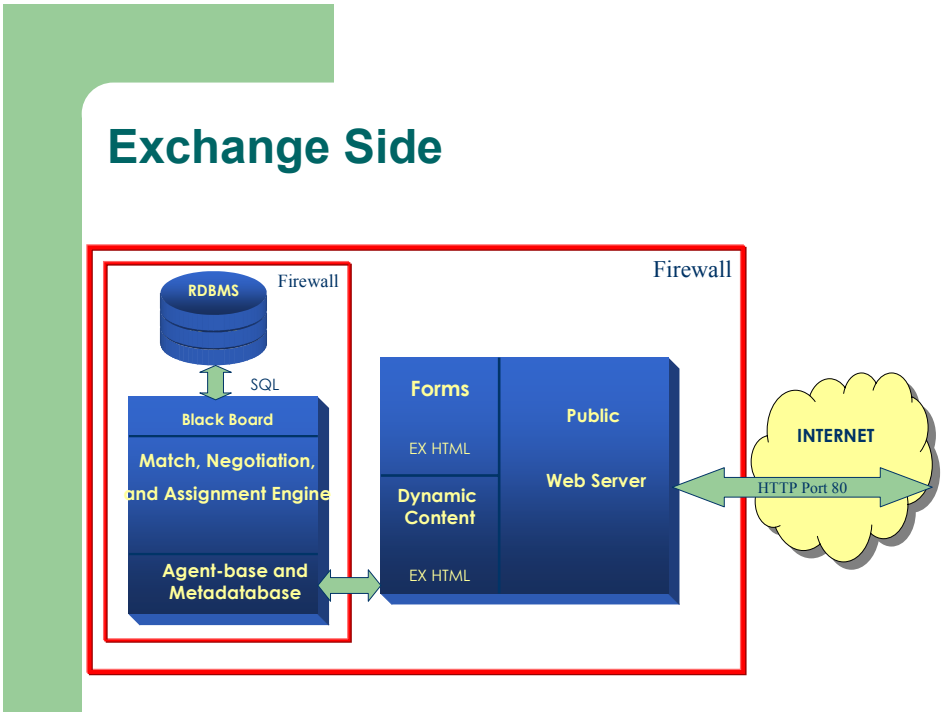

Figure 3, the Basic Structure of the Exchange Side.

When the enterprise resources concerned are focused on databases, the architecture can be elaborated for the participatory database query problem with particular execution methods to implement the two-stage solution. We also need implementation strategies, such as how an information enterprise might map the pricing model to its organizational valuation metrics and budgets, and what standards it might adopt to inter-operate the artificial market with other functions it conducts. We treat these topics in the next section.

4. The Two-Stage Collaboration Methods for Participatory Global Query

The Two-Stage Collaboration solution to the participatory database query problem uses the above ERM model. The major elements of the model are developed below. Each element is also discussed as an element of the general solution to the broad enterprise resource allocation problem wherever appropriate. Some of the key methods and techniques for the second stage solution are adopted, with extensions, from our past research on global query systems, especially the MQL (Metadata Query Language) algorithms [Cheung and Hsu 1996] and the distributed ROPE (Rule-Oriented Programming Environment) shells [Babin and Hsu 1996]. The elements for the first stage solution are new, although they also leverage the past results [Hsu et.al. 1991].

4.1. The Agent Model: Task Agent, Agent-Base, and Metadatabase

Resource providers and users initiate their offers and requests through custom-created **task agents**. They log on (remotely) to the Agent-Base at the exchange site and instruct it to create a task agent for their offers or requests; and get the software agents uploaded to

their respective local sites after creation. They use the same mechanism to update or delete their agents. The participants can now initiate a task by launching the agent created to the Black Board of the exchange site. The agent ***publishes*** its information content (see below) at the Black Board, with possible subsequent modifications. The Black Board uses this information to conduct matching, negotiation-auction, and assignment, as described in the next section on Black Board. The agent has three basic elements:

Define *Agent* (**C**ommunicator, **I**nformation **C**ontent, **R**ule-base*)*.

The *communicator* includes a header (e.g., ID or IP address, XML-SQL or inter-operation protocols) and other metadata and routines required for agent processing. The field offers many good practices which can be tailored for the user community according to the operating policies of the enterprise to design the communicator. The *information content* describes the conditions and semantics of the task according to certain representation methods acceptable to the enterprise. In any design, the conditions specify the price demanded or offered, the deadline, and processing requirements; while the semantics use either the common schema of the organization or the common dictionary of keywords to communicate to other agents the information nature of the task. The processing requirements include the type of resource offered or requested, job constraints and task status if the task belongs to an automated workflow (series of single tasks) or complex task. A complex task will be processed as a sequence of single tasks according to processing or workflow rules. The *rule-base* contains operating knowledge for conducting automatic negotiation, auction, and other similar behaviors, such as choices of pre-determined negotiation schemes. It also contains workflow rules and other logic for the processing of complex tasks. Its representation follows directly the design established in [Bouziane and Hsu 1997]. All agents follow a unified protocol regulated by the Agent-Base, which is a shell operating on top of a repository of enterprise metadata and other knowledge, called Metadatabase, to create and manage agents for the market.

The **Metadatabase** employs a particular representation method, TSER (Two-Stage Entity-Relationship) [Hsu, et.al. 1991], to integrate enterprise information models, contextual knowledge, software resources (for inter-operation), and user-application families. These enterprise metadata are structured into a database so that the community can query, manage, and evolve enterprise metadata through the Metadatabase for their tasks in the same manner as they could with a regular database. Therefore, the Metadatabase is a repository of enterprise global query policies pertaining to the artificial market (such as rules about particular user-application families, entities, and relationships), as well as one for participant data models. Moreover, the scope of the representation method covers all three elements of the software agents; thus, the Metadatabase can also be a depot of re-usable objects or common raw materials (communication software, information content, and rules) from which the Agent-Base builds task agents. The model also supports some classes of natural language interface using user-defined extensible keywords [Boonjing and Hsu 2003]. In fact, the Metadatabase amounts to a version of the communal common schema for the (extended) organization (see Section 4.6). It, along with the proxy server (see Section 4.3), constitute the connector through which a participating system plugs into the overall artificial

market. While the proxy server provides physical connection in an API manner, the Metadatabase offers logical integration for the enterprise.

The MQL, Metadata Query Language, provides a means to define the Information Content of an agent for global query. The additional elements of the IC become extensions to the MQL. Basically, they will take a declarative form and provide pointers to the software objects contained in Rule-Base. An illustrative example looks like this:

EX1: *Get* customer *where* make = 'Ford' and product = 'mustang' *using match* "2" *auction* "3" *with price* = "30" *and deadline* = "1 hr".

The user specifies the matching rule to be within 2 percent of the price and auction the 3rd regime, as provided in the extended MQL. The query part of the IC, i.e., the Get and Where clauses, can also be expressed in a free text natural language form:

EX2: "Show me who have bought Ford Mustangs."

The provision of information can use designated commands such as "*Provide*" and "*Have*" in a similar way. The syntax of the Extended Metadata Query Language is beyond the scope of the paper. The properties and limitations of the metadata-based natural language query are available from the literature [Boonjing and Hsu, 2002]. In any case, the templates or canned methods that matching, auction, and complex tasks use can be included as enterprise metadata in the Metadatabase and be copied into the Rule-Base of the agent to make agents self-reliant in distributed computing at local databases.

The **Agent-Base** is primarily a method to mass-customize large amount of agents at run time. Since the agents could number in thousands or even millions at any one time, and be custom built, the new model needs an efficient way to create and manage these agents online and on demand. The Metadatabase provides community resources required by mass production, while the Agent-Base customizes the configuration of these resources for particular tasks. It will also support the owners of the agents to add information (e.g., specific data values of some entities, relationships, or attributes, and operating rules) and route them to the Metadatabase for possible inclusion into its content. Another aspect is the ability to automatically update the contents of agents when these metadata are changed at the Metadatabase. This capability is very important for maintaining the logical consistency across agents, or the integrity of the agent community. The third aspect is a log of the task agents currently active at the Black Board. This design allows the software agent to be a persistent surrogate (24/7) at the market for conducting asynchronous negotiation, among other things (first stage solution). Thus, the Agent-Base is both a management shell and a gathering of active agents.

4.2. The Black Board: Match, Auction-Negotiation, and Assignment

The Black Board serves the agents and conducts match, auction-negotiation, and assignment during the first stage. It then administers the global query algorithm during the second stage. This engine maintains a list of all tasks published by agents, including

their information contents, and consults with the Metadatabase for the latest enterprise policies to avoid chaos at the artificial market. One of these responsibilities is to break ties and ascertain that all (worthy) tasks find a match before their deadlines.

The basic logic of match goes in this manner. For a given task, it first satisfies the semantic constraints by looking for counterparts possessing the same information content. The match can either be exact or partial, depending on the rules specified in the IC of the software agent. When semantic constraints are met, the matching proceeds to check conditions. If single perfect match exists, then the Black Board will assign the task to the resources (databases) matched. If multiple perfect matches are found, then a round of auction among them will decide the final assignment. If only partial matches are found, then the task enters negotiation. The negotiation could be automated where the agents at the agent-base use their rules to find an optimal match and break ties by auction; or, the agents could inform the task initiators to modify the original conditions – i.e., human intervened negotiation. On the other hand, if no match, perfect or partial, is found, then the agent and/or the initiator could update the task information content depending on whether the difficulty is caused by semantics or by conditions. The Black Board will post current market conditions – e.g., statistics of bids, usage patterns of keywords, and status of hot issues – to facilitate the modification and negotiation. The initiators could also proactively update the task agents stored at the local site and re-launch it to replace the old one. Certain conditions, especially those related to workload at local resources, would be suitable for automatic update from the participant sites. The Black Board intervenes only on an as-needed, exceptional basis to break ties and enforce enterprise policies. For example, it could check on certain types of requesting tasks that have no match and increase their offering prices on a loan basis to find them a match on or near the deadline.

The assignment phase is essentially a connection (notification) between the seeking tasks and their satisfying resources (databases). It entails an update of the communicator of the resident agent by the Agent-Base, if necessary, to prepare it to communicate with the proxy server at the destination site. The updated agent will then be uploaded to the task initiating site and initiate a peer-to-peer transaction from there. Now, the task agent is ready to ***subscribe*** to the resources; namely, connect to their system. This subscription triggers the second stage solution of the participatory database query. The Black Board at the new stage interprets the global query (in MQL, as in this design) requested by the task agents and decomposes (if necessary) and translates it into local bound (sub-)queries in the local language (SQL) according to the metadata about the local databases maintained in the Metadatabase (software resources and data-entity-relationship mappings). These local bound (sub-)queries are then transmitted to the ROPE shells at the local proxy servers for distributed execution. Results are sent back and assembled at the Black Board, which in turn returns them to the requesting sites and updates or removes the task agents for both the demand side and the supply side upon completion of the global query job.

The two-stage solution to participatory database query has a complexity dominated by the first stage, whose analytic nature is **large-scale, real time scheduling** (of databases to queries). It is well known that the computational complexity of traditional scheduling algorithms (global control) is NP-hard; while the complexity of sorting (according to

price) is *O(*NlogN*)* with N being the number of tasks. Thus, the self-scheduling nature of the ERM assures an efficient regime of computation with complexity in linear to low order of polynomial (including negotiation and feedback), and makes the solution scalable exponentially in theory. As discussed before, the Black Board administers a global query algorithm such as the one provided in the Metadatabase model [Cheung and Hsu 1996] at the second stage once the participation of databases is determined by the assignment. The processing of the global query algorithm is concurrent in nature through a distributed shell, called ROPE [Babin and Hsu 1996], in the proxy server at the local databases. Thus, the complexity is inherently linear to the number of tasks involved, too.

4.3. The Proxy Server: Peer-to-Peer Transaction and Systems Inter-operation

The proxy server is a software system that the artificial market places at the local site. It accomplishes two basic jobs towards connecting the exchange side with the participant side: systems inter-operation and peer-to-peer transaction. The server interacts with the exchange site and collaborates with the Agent-Base to store, maintain, and process (launch) the software agents owned by its local site, as well as to respond (execute) to the call of task agents from other sites. In this capacity, it functions as the server of the local site for all task agents initiated at the site but processed elsewhere at the Black Board or other local sites. Thus, it executes the workflow logic for its complex tasks to, e.g., sequentially launch the component single tasks and maintain the overall task status. The server offers a data standard for the task agents to communicate between themselves. The standard is maintained together with the Communicator design of the agent. The exact standard will depend on the enterprise requirements; but a good, standing design is to use XML-SQL, coupled with the MQL. That is, the proxy server has a ROPE shell and a standard protocol to receive and process task agents for information transfer. Part of the protocol is a standard schema for view tables contained in the ROPE shell that the proxy server maintains for interfacing with the local databases. It works two ways. First, the local database publishes the select content that it slates to share with the artificial market as view tables at the proxy server, using the format provided by the latter. The information requesting task agents that the Black Board sent to the local database can then query the view tables to retrieve or transfer the selected content in the following manner: The agent transmits the query to the ROPE shell, which posts the query against the view tables for the host database to execute. Second, if the proxy server's own task agents transfer in content from other resource sites, then the input is stored as view tables for the database to acquire under its own management. In either case, the ROPE shell enables the mobile agents (the ones being sent by the Black Board) to inter-operate with local databases without breaching the local sites. The queue discipline at the proxy server is self-scheduling based on prices. However, if the requesting task from outside is to use the computing facility for data processing, as opposed to information retrieval, then the proxy server passes it as a regular job to the local system and follows the local queue discipline. In a similar way, it monitors the work load, task status, and other relevant data of the host server to update its task agents at the Agent-Base and elsewhere in the system.

The proxy server also controls peer-to-peer negotiation, including matching and auction, as described in the next sub-section, if the local site has sufficient computing power to

invoke this option. In this capacity, the proxy server launches its task agents to visit task agents published on proxy servers at other local sites, as well as prepares them for negotiation with visiting agents from other participants. The proxy server from which the task agent first (time stamp) initiates a peer-to-peer negotiation in the community will function as a ***mini Black Board*** during the life of the negotiation. The basic logic of the Black Board applies here, except that the initiating task agents call on other sites rather than other agents posting onto the initiating proxy servers. As such, a proxy server might control several concurrent auctions involving different tasks at different sites, and the community might have numerous such auctions controlled by different proxy servers at numerous local sites at the same time; all are autonomous to the global Black Board at the market site. In this mode, a proxy server will maintain a list of visiting task agents and matches its own task agents against them.

The employment of proxy servers allows for peer-to-peer transaction and hence concludes the assurance of the computational efficiency promised by the price-based Black Board. Peer-to-peer transaction is advantageous to the environment for these basic reasons: (1) it allows for the participants' direct control of all tasks originating at their site as well as tasks being processed there, and hence simplifies the global control requirements and work load; (2) it supports distributed updates and processing of agents based on local conditions; and (3) it provides a backup to the Black Board. Furthermore, it also contributes an open and scalable way to connect any number of local databases and other resources into the Enterprise Resources Market without interrupting the operation of the market. Along with the Metadatabase, which offers an open and scalable way to incorporate any number of information models into the market during the run time, and the Black Board, which promises computational scalability, these three elements make the proposed ERM design uniquely open and scalable.

4.4. Peer-to-Peer Negotiation: Virtual and Distributed Mini Black Boards

Peer-to-peer computing has a general complexity of $O(N^2)$ which hinders scaling-up. We develop a new basic logical structure, the **match circle**, to reduce the complexity. The circle recognizes the group of nodes (local sites) whose tasks match. Pair-wise computing is unnecessary within a match circle since it has a node serving as its **mini-Black Board**. A local site can have a number of simultaneous tasks alive in the community and hence can belong to a number of simultaneous match circles. Therefore, both the circles and the mini-Black Boards are virtual and task-based. The overall complexity of the peer-to-peer computing is primarily $C*O(N)$, where C ($<<$ N when N is large) is the number of such virtual circles and $O(N)$ is the complexity of distributed computing within a circle.

The Agent Base maintains a global protocol for determining time stamps for all task agents initiating a negotiation, which starts with a search for matches at other local sites and finishes when an assignment is finalized after, if necessary, auctioning among multiple matches. The negotiation at the matching phase is concerned with task constraints, while it is about the objective function when auctioning. The initiating task agent, the one with the earliest time stamp among all agents that matched, has the control of the negotiation and its proxy server becomes the mini-Black Board. It first launches a

round of search where the task agent visits and looks for matches at (all) other local sites in the community. If a single perfect match is found within a pre-determined time period, then the proxy server acts according to the nature of the task: for requesting task, it obtains necessary inter-operation parameters or routines from the Agent-Base for the task agent and sends it to queue at the matching site through the proxy server at that site; and for offering task, it informs the proxy server of the match task to launch the requesting agent. Optionally, the proxy server could also contain a reduced copy of the Metadatabase to allow it to augment its task agents with the inter-operation data. If multiple matches exist, then the proxy server conducts an auction among the matches where it singly controls the auction session. The result will either be a single meeting of the best price – in which case the proxy server assigns the task as mentioned above, or a declaration of failure of the auction which results in a deletion of the current task. The participant in the latter case can opt to re-initiate the task or re-create a new task agent.

If no perfect match is found when the time expires, then the initiating task agent every where starts a round of lock-step negotiation with its host agents, respectively. Each pair of negotiation is independent of all other pairs under the initiating task agent's autonomous control and uses the same negotiating regime. The regime could be rule-based, staged modules, or any appropriate design, as long as it follows a definitive procedure to define and control the gives and takes, with each step associated with a certain time window. Thus, all simultaneous negotiations at all local sites proceed in the same step by step at all times. Each step modifies certain constraints in certain manner within each window, and the proxy server terminates the negotiation at the first moment when a perfect match or matches are found. At the conclusion of each step of the modification on constraints, the task agent returns the matches (when achieved) with an indicator of the step during which they are obtained, along with the identification of the local proxy servers of the matched task agents. The initiating proxy server could use the indicator to prioritize the matches in the assumption that more modifications result in less favorable terms and hence the earlier the matches the better. This way, if some successful negotiations are reported back late because of their local queuing situations, or any other reason, they could preempt incumbent matches or ongoing auctions when they arrive at the initiating proxy server. Alternatively, the proxy server could opt to ignore late results. The task agents use the time-based progression of negotiation to self-synchronize their autonomous processing. The initiating proxy server, i.e., the mini Black Board, does not actively control the processing of its task agent at each local site, but only determine the matches from all reported results. When the time expires without any matches found, the task agent ceases to exist and the owner could either revive it or forgo it. The peer-to-peer design allows a proxy server to control the negotiations (matching and auctioning) of its task agents, individually as well as collectively (to manage its own local resources), and thereby promotes distributed computing. There could be many proxy servers controlling many concurrent match circles each of which represents a virtual information shop.

The second stage solution requires the proxy server to possess a copy of the global query system, as discussed in Section 2 for the global Black Board. The MQL algorithms always include the capability to administer concurrent ROPE sessions, as required for supporting concurrent mini-Black Boards. The ROPE shell everywhere has the same

design and the same metadata definition, and hence can inter-operate with each other in the same time. The limiting factor of the concurrency is basically the capacity of the proxy server to handle it at the local database site.

4.5. Implementation Model: Organizational Metrics and Data Standard

To implement the artificial market in an enterprise, we need to map the pricing/rewarding model to organizational control metrics and incorporate the ERM elements into the organizational systems. We assume that the (extended) organization uses a budget model to control the overall allocation of resources to operating units and individuals. The model will create artificial funds (money or fungible credit) for participants of the Market, and periodically deposit or adjust them according to their overall performance at the market. In addition, the market maintains these funds and will automatically adjust them for the participants after each transaction to reflect their revenues (sales) or payments (purchases) in a manner similar to a bank. Thus, individuals and operating units do not exchange money directly, but their purchasing power at the market. The bids, therefore, reflect the perceived value of the resources requested or offered by the participants. To make the scheme work, the fund owners must have control of their funds and the ability to use surplus for real world purposes such as hiring people or purchasing facilities. Thus, the market itself is the first mechanism to measure performance, reward the participants, and thereby reallocate resources. The managers of the participants will also assess the performance of resources in terms of value added to their missions; that is, the quality and quantity of information provided and utilized. They could measure the value in terms of how many people have used and benefited from the offerings (sales) and how actively people seek out for useful information. The system will generate market statistics on problems (e.g., loans), requests, offerings, transactions, connections, keyword usage (hot topics), as well as accounting and scheduling logs. Results are performance feedback to the managers for the budgeting process and reward systems. This periodic review assisted by market statistics is the second line of control and reward for resource allocation.

The actual pricing model has to be determined case by case for the application concerned. The implementation should also consider the possible collaboration of the Market with other enterprise management functions. For instance, the patterns of subscription (connection) reveal the need for new or ad hoc channels of communication or workflow processes. Thus, these data could feed into such models as organizational networks and processes for the evaluation of the (extended) organization.

The Metadatabase model provides a structural way to incorporate the enterprise databases into the Market, by using the ROPE shells, organizational metadata, and a common schema in the form of a Metadatabase; as discussed above. However, additional data standards would still be necessary in actual implementation. For instance, the MQL needs to map to local data languages and the ROPE shell to the enterprise middle-ware; therefore, standards on these enterprise elements are needed. We are confident that one can adopt the best practices in the field of e-business and database integration to satisfy

the need. Generic technologies such as JAVA, SQL, and XML could be employed for coding, and MQL and ROPE could be coupled with Corba and J2EE, too.

4.6. Open Common Schema: a Metadatabase for Extensible Information Integration

All artificial markets face the pivotal challenge of how to process heterogeneous data semantics unless they can impose a single, comprehensive standard on all participants. The section on the Agent Model provides a Metadatabase to represent data semantics of all resources in the enterprise. This task usually corresponds to developing a common schema at the high end of integration, or a reference dictionary of keywords at the low end. Current practices in the field at both ends have certain drawbacks: available common schema methods tend to be hard to develop and too rigid to maintain, while keywords might not cover the full semantics contained in information models (especially relationships and contextual knowledge - processes). An alternative is to base the common schema on ontology - a basic structure of semantics at a meta-level, rather than on tenuous instances of practice. The Metadatabase model is such a design.

The ontology it uses is constructed from generic information modeling concepts per se; i.e., it is methodology-based. The ontology, based on the TSER method, is comparable in concept to the Information Resources Dictionary System (IRDS) effort of NIST [Hsu 1991] and similar approaches in the field. However, it differs fundamentally from application-based ontology that seeks to capture the basic logic of an entire application domain. The methodology-based ontology offers efficiency and simplicity (minimalism); however, it is limited to the applicability of the method on which the design is based. The TSER, Two-Stage Entity-Relationship, model is an extended entity-relationship-attribute model shown to be scalable (i.e., metadata independence) [Hsu 1996] and capable of incorporating rules {Bouziane and Hsu, 1997], as well as to be applicable to a range of application databases commonly involved in global query processing.

The icons of the Metadatabase structure, or the graphical representation of the ontology, shown in Figure 4, represent either a table of metadata or a particular type of integrity control rule. The ontology in Figure 4 extends slightly the previous Metadatabase structure by also including user words and cases (as in case-based reasoning), to support agent definition. The metadata include subjects and views, entity-relationship models, contextual knowledge in the form of rules, application and user definitions, database definitions and database objects. User-words are defined as ordered pairs (class, object). Classes include *Applications*, *Subjects*, *EntRels* (entity-relationship), *Items*, *Values*, and *Operators*; all of which are metadata tables as shown in Figure 4. Objects are instances (contents) of these classes. An object is uniquely identified by an ordered quadruple (Item name, EntRel name, Subject name, Application name) as well as an identifier. A case, useful for rule-base, consists of a problem definition and a solution, but not the usual outcome, because the Metadatabase contains the complete domain knowledge needed. New problems (e.g., exceptions to general policies) would use the problem definition to find the (best) matching cases and apply the associated solutions to them. A set of metadata for a task describes the problem definition, and its interpretation defines the solution. Cases also strengthen the Black Board's ability to perform real time matching

and assignment of tasks when uncertainty arises. As such, the Metadatabase collects local information models as the elements (metadata entries) constituting the common schema.

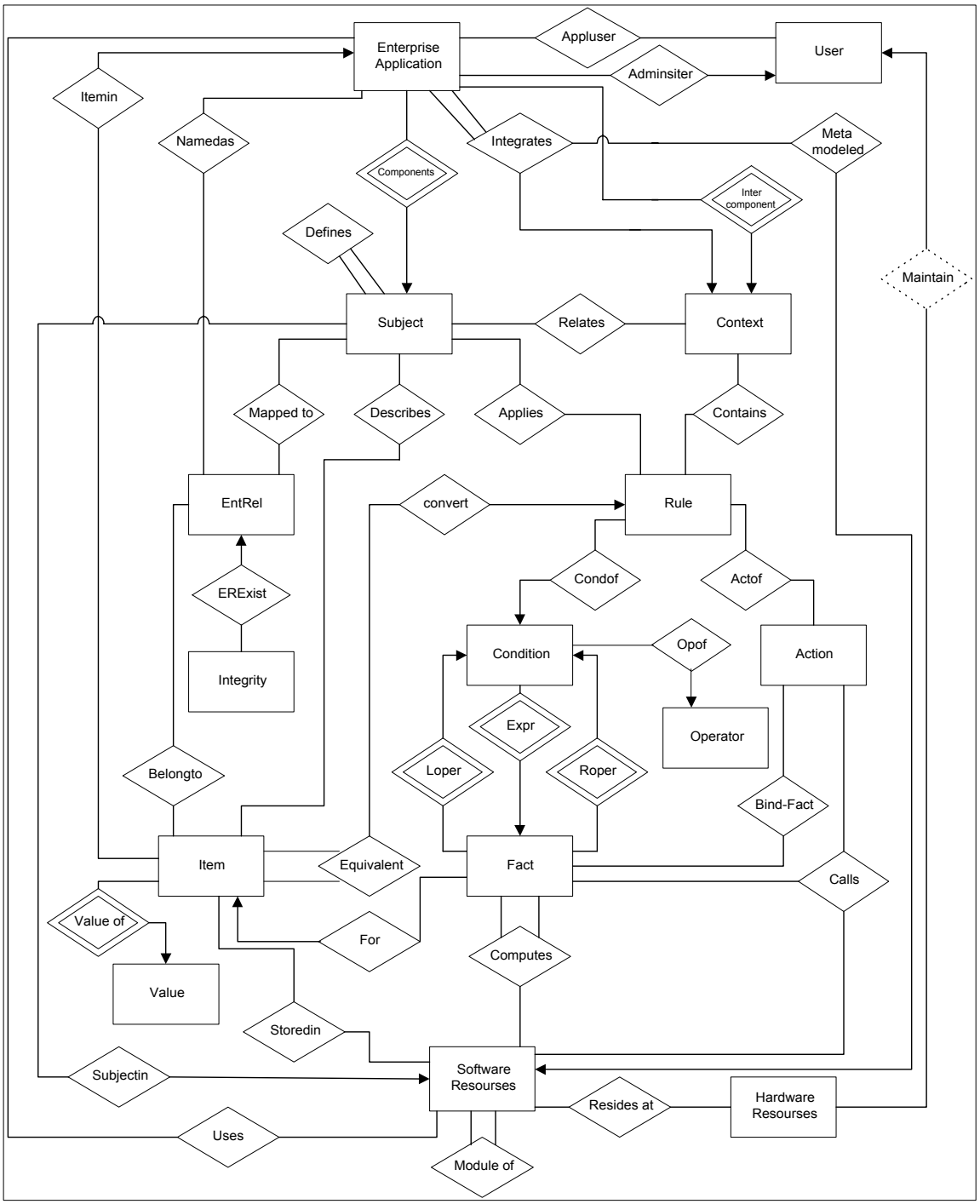

Figure 4: The Ontology: the structure of the Metadatabase.

The common schema so constructed, as a Metadatabase, will be able to accommodate changes, including deletion, addition, and modification of information models for local resources. For instance, when a new local database is added to the Market, the local site will register itself and create an information model using the TSER methodology to define the database. The information model will then be added as new metadata entries to the appropriate meta-tables of the Metadatabase (similar to SQL Inserts). This process is amenable to automation using a CASE tool. The Metadatabase does not need to shut down at any time during the update, since the operation is really a regular database job. After this logical connection, the Market will install a proxy server fine-tuned for the new database in the local environment. This installment does not interfere with the regular operation of the (rest of the) Market, and hence the whole addition process will not affect any existing local systems nor the on-going tasks at the Market. Once the process is completed, the new database takes part immediately and automatically in the Market. Other changes are similarly self-contained and autonomous. Therefore, the design offers an open common schema to enable extensible information integration for the community; i.e., help make the Market open and scalable.

The Metadatabase has been tested extensively in LAN, WAN, and even Internet-based environments at some industrial companies. However, it has not been applied for a participatory community. Thus, its development into the common schema represents a new contribution to the filed.

Among the six elements discussed above, the Black Board and the Proxy Server are synthesized from results in the field, and the Implementation Model is organization-specific. The other three, namely, the new Agent Model, Peer-to-Peer Negotiation, and Open Common Schema, are new contributions of the paper. Together, they constitute a complete design for the two-stage collaboration solution to the participatory database query problem. We developed a preliminary prototype to help reduce the concept to practice and test its performance concerning, among other things, the potential bottleneck in the Black Board. The next section describes this laboratory test.

5. The Preliminary Prototype: A Match Engine

The participatory database query problem leverages existing query methods for its solution at the second stage. The first stage solution, however, requires the Blackboard to perform massive concurrent matches (see Section 4.2), which is a new challenge. A key to the feasibility of the Enterprise Resources Market model is, therefore, the feasibility of constructing such a match engine. A complete agent system could use many established results in the field. However, since our goal is rapid prototyping to test large-scale concurrent matching, we did not focus on agents per se but on the match engine, and employed the best practice available to our needs.

When considering the development of this type of system, several requirements of the software were immediately observed. First, the match engine must have a high degree of availability. This means that the matching and negotiation should only be offline in the event of some catastrophic failure associated with the underlying hardware and infrastructure. The software should be as fault resistance as possible. Second, the

matching engine must be extremely efficient. Last, the matching engine must communicate to potentially many different third-party software systems as well as provide seamless communication and coordination with other possible exchanges or matching engines in the (extended) community.

On the software engineering side, there were a number of requirements as well. First, the engine should be easily developed using existing software libraries and not require developing our own low-level system subcomponent and libraries. That is to say, one did not want to re-write / develop the primitives, such as "threads". The development environment should yield highly efficient code for the underlying matching algorithms as well as not require one to re-write or develop separately specialized software for external connections to third-party software systems, such as an Oracle database. Last, we needed a rapid prototyping environment for obvious reasons.

Given these requirements, the safe choice seems to be Java. After all, Java supports much of the library and core functionality. However, there are two core requirements not readily addressed by Java. First, Java's garbage collection engine has the potential to thrash when memory on long lived objects becomes tight. Consequently, predictable execution over long periods of time may not be possible and thus the high-availability requirement would not have been met. Next, it is problematic for one to "hot swap" code class modules at run time. In Java, old objects would use the old methods and not the new methods specified by the new class module. Thus, to perform a simple update of the system, the entire exchange would need to be taken offline. This was unacceptable given our availability requirements. Finally, Java fails to provide an efficient representation required of the matching algorithms. Thus, we would not be able to meet the execution-time requirements if we were to use Java.

After an investigation of programming languages and environments, we decided on Erlang (www.erlang.org). Erlang is a functional programming language designed at Ericsson Labs for use in telecommunication systems. In particular, Erlang is the language used to implement the routing software in one of Ericsson's high-speed ATM switches. This capability could never be accomplished in Java. The core language provides a number of features, many of which gave it the ability to meet our system and software engineering requirements at the time of development. These include: ultra-light weight threads for concurrency, error detection methods for recovery and robustness, software real-time scheduling, predictable incremental garbage collection, "hot swappable" and "incrementally loaded" code modules, highly efficient code representation, and external interfaces to other third-party software.

Because Erlang is functional, data are passed as arguments to functions. Even more interesting is the fact that all variables are single assignment. So, here a variable can only be written once. This implies that when a piece of code is "hot swapped" you need not ever worry about "old" objects; instead a data item will always be operated on by the most recently loaded version of any given function. Single-assignment variables also greatly simplify garbage collection, thus allowing it to be accomplished in a very predictable manner which greatly increases system availability and robustness.

The matching engine, written in Erlang, is coupled with a DBMS (database management system) to provide an overall set of services for performing a transaction at the Black Board. In this reduced prototype and simulated testing, when a new request arrives a new agent is spawned. Agents are realized by creating an Erlang thread. Once created, that thread is allowed to search the DBMS for any potential matching entries. If a match is found, each matching entry is sent to a sleeping thread. Once the entry (data) and the thread are bound together, they form a run time agent. These agents, using Erlang message passing primitive, schedule "provide" or "request" events to each other. Agents discover each other via the DMBS search process. Having received an event, an agent can use its internal evaluation function to determine the goodness of the fit. Because this evaluation function is specified in Erlang hot-swappable, the end employer of the agent can dynamically update it on the fly (i.e., install a new function). This gives the end users of the system a wide range of flexibility in how their agents negotiate. One simple example of an evaluation function is based on price and time. If the provider is willing to pay a price within X percents of the requester's price, then this is considered a match. Similarly, the provider agent can use time to either increase or decrease the price. So, for example, if the provider's information is time-sensitive, or "perishable", then it might elect to have its agent drop the price steadily over time.

After having found a match, the agents report back to their end users that a successful transaction has taken place. The match is logged into the DBMS. By logging the transaction in a highly reliable DBMS, we prevent non-repudiation of completed transactions. When a match was not found, the agent (realized as an Erlang thread) is then put to sleep and the request data is placed inside the DMBS, making it available for searching by other agents. Please note that as an agent is given a "turn", it has the option to modify its parameters which determine a match. For example, it could lower its (asking or bidding) price or changing other terms in order to find a match.

The matching and negotiation engine resides on an isolated local-area network at the market site behind a web server and firewall, thus, a high-degree of security is provided in this design. Only the exchange administer has direct access to the exchange system. As for scalability, we find that our initial prototype would scale up to10,000 agents (i.e., threads attempting to match requests) under concurrency control, and up to a million without it. However, we discovered that Erlang's DMBS engine called Menesia presents a performance limit. To reduce this bottleneck we could migrate to an external DBMS engine such as Oracle. There, we anticipate that it be able to scale up to 100,000 agents per machine, which is an upper bound on the number of threads Erlang can handle. We note however, that this limitation can be overcome by using the thread migration capability within Erlang. Here, up to 256 machines can be directly connected under the Erlang runtime environment. Thus, the current design could provide the seamless ability to scale up to millions of agents across a distributed network in the Exchange Domain.

6. Conclusion: From Databases to Enterprise Resources

We have developed a Two-Stage Collaboration solution (Sections 4 and 5) to the

participatory database query problem (Section 1) using an Enterprise Resources Market model. The ERM model is also presented in a broad context of the enterprise resource allocation problem (Sections 2 and 3). We submit that the participatory database query problem is a core to the general problem and represents a high end of integration for extended enterprises.

The main conceptual foundation of this work is the definition that participatory global query has two sub-problems: participatory database allocation and query processing after allocation. The first problem, self-allocation, goes beyond the traditional database query literature. The analysis progresses to realize that an industrial exchange - an artificial market - is a basic self-scheduling mechanism with the promises to maximize the overall value of resources while satisfying tasks' requirements. That is, a market can optimize the assignment of databases for queries for very large scale information enterprises. However, previous market models did not include sufficient methods for global database query processing. Therefore, we synthesize traditional database query with artificial market to develop a complete design and solution for the participatory database query problem – i.e., integrating the new Enterprise Resources Market model with the previous Metadatabase model to result in the Two-Stage Collaboration solution. New results include an agent model, a peer-to-peer negotiation design, and an open common schema method; which in their own right contribute respectively to the software agent literature, e-auction, and the field of industrial exchange. A prototype match engine helps reduce the concept to practice and verify the computing feasibility of the ERM model. Preliminary results of the testing in a laboratory environment have shown that a match engine, which is a potential bottleneck of computing for the whole approach, can handle a large amount of simultaneous task agents while running on a relatively simple platform using only off the shelf technology.

The above results contribute to two general problems: real time global resource allocation and information integration for extended enterprises; both of which are hard and yet critical to IT-based organizations in today's digital society. Its contribution in the real-time scheduling areas is its offering of a true market mechanism to provide comprehensive performance evaluation and feedback and thereby achieve optimality in self-allocation. For information integration, the work ***brings the capability of global database query to industrial exchanges***; which are an important mechanism to achieve extended enterprises. The previous exchanges tend to limit the data transactions to the processing of business documents and straightforward file transfer. More specifically, the agent-base method helps make large-scale agent-based systems more manageable. Although industrial exchange models have always called for the use of agent technology, they tend to stop short in practice because of insufficient capacity for managing large-scale agent community. The concept of mini-blackboard helps enable a distributed, concurrent, and virtual peer-to-peer community formed around tasks and/or bids, as required in e-auction for exchanges.

We submit that the basic logic of the Two-Stage Collaboration solution is applicable to the general enterprise resource allocation problem beyond databases per se. The solution starts with a market design – not just a market metaphor - that satisfies the requirements

of information enterprises. The design provides self-scheduling to balance demand and supply, with computational efficiency (linear to low polynomial complexity) and the capability of information interchange (semantics match) and self-correction (performance feedback and reward), generic to information enterprises. Of the solution, the ERM model extends the previous exchanges from focusing on bids to also allocating resources for actual execution; and from relying on a global black board to also allowing peer-to-peer processing. The new agent model and agent-based architecture connect large number of distributed resource providers and users, and allow them to use software surrogates to globally publish their offers and requests of resources and to locally subscribe to the resources. Both a traditional global Black Board and a new design of distributed virtual mini-Black Boards for peer-to-peer transaction are included. A pricing/rewarding model ***encourages*** sharing of information resources in the same time as it controls the use of them. In this regime, both providers and users can initiate tasks as bids for transaction, while an agent-base facilitates the creation, management, and processing of their custom (task-oriented) agents. Overall, the solution uses metadata technology to represent task characteristics, including transaction requirements and data semantics, and to match requests with offerings according to these characteristics. As such, the ***objective function is the maximization of the total (perceived) value of information and physical resources***, where the valuation is the pricing/rewarding model. This objective function encompasses and synthesizes such criteria as minimization of delay in assignment and optimal utilization of resources, as well as provides performance-based rewards and adjustment, in a way similar to that associated with a natural market. ***The constraints are the transaction requirements and data semantics of tasks***, which could be updated real-time and online through the task agents, based on local conditions (e.g., work load and deadline). When a match of demand and supply bids are found, an optimal allocation for the tasks at hand is identified – this is the first stage solution. The problem now reduces to one comparable to a traditional transaction regime for this particular instance (matched tasks). The execution can now focus on the task(s) to trigger processing at the local sites that bound themselves to participate in the task(s), and obtain answers. This is the second stage solution. The market allows the many tasks that could take place concurrently to coordinate themselves through sending feedbacks by agents and proxy servers to the global blackboard or the mini-blackboard. Self-scheduling takes place at matching, the allocation-connection, and the queuing at the local resources; and hence assure computational efficiency through distributed processing. We should note that the market still matches tasks and helps interchange of information resources even without a price (reward) model; the difference is the optimality of the assignment. Finally, it seems that a general publish and subscribe model in the form of an ERM resonates well with digital society, as information is central and universal in such a society, and the concept of market far reaching.

One immediate area for future research is concerned with the requirement of a common schema. The proposed solution requires a participating database to register with the common schema, which is based on the Metadatabase model. This represents a limitation to the openness of the community; although the Metadatabase model is shown to be relatively open and scalable for accommodating many data models. The limitation might

not be justifiable for many applications outside the scope of an enterprise or an extended enterprise. In the ongoing work, we will explore the possibility to further open up the common schema or to trade functionality for easy participation. We will also explore the possibility of reducing the scale of resources from databases to elements of databases or to computing resources. Along this line, works could explore the value of applying the market approach to single database query and to the notion of the economics of computing. Another direction of further research is to particularize the general solution for other specific classes of information resources, as well as seek a more efficient and effective overall design. A third direction would be the experimental or even empirical investigation of the market approach vis-à-vis traditional approaches for solving some hard problems in real-time scheduling. The following criteria will be suitable for this investigation: the global valuation of resources allocation, the performance of scheduling on individual tasks (timeliness), the accuracy of tasks matching, and the potential benefits to information enterprise environments. The last criterion could expand into, e.g., how the market might help connect field officers and inter-operate intelligence databases across organizations. It will also examine how software agents' ability to allow for anonymous publications of tasks (both resources providers and user) and subscription of resources helps the information enterprises.

References

G. Babin and C. Hsu, "Decomposition of Knowledge for Concurrent Processing," *IEEE Transactions on Knowledge and Data Engineering,* vol. 8, no. 5, 1996, pp 758-772.

A. Baker, "Metaphor or Reality: a Case Study Where Agents Bid with Actual Costs to Schedule a Factory," in Market-Based Control: a Paradigm for Distributed Resources Allocations, ed. S. Clearwater, World Scientific Publishing, River Edge NJ, 1996.

A. Baker, "A Survey of Factory Control Algorithms That Can Be Implemented in a Multi-Agent Hierarchy: Dispatching, Scheduling, and Pull," *Journal of Manufacturing Systems*, Vol. 17, No. 4, 1998, pp. 297-320.

V. Boonjing and C. Hsu, "Natural Language Interaction Using a Scalable Reference Dictionary," *Proceedings of the 8th International Conference on Applications of Natural Language to Information Systems,* Burg, Germany, June 2003.

M. Bouziane and C. Hsu, "A Rulebase Management System Using Conceptual Modeling," *J. Artificial Intelligence Tools*, Vol. 6, No.1, March 1997, pp. 37-61.

R. Braumandl, M. Keidl, A. Kemper, D. Kossmann, A. Kreutz, S. Seltzsam, and K. Stocker, "ObjectGlobe: Ubiquitous Query Processing on the Internet," The VLDB Journal, Vol. 10, 2001, pp.48-71.

A. Cesta, A. Oddi and S.F. Smith, "Iterative Flattening: A Scalable Method for Solving Multi-Capacity Scheduling Problems", *Proceedings National Conference on Artificial Intelligence (AAAI-00)*, Austin, TX, July, 2000.

W. Cheung and C. Hsu, "The Model-Assisted Global Query System for Multiple databases in Distributed Enterprises," *ACM Transactions on Information Systems,* vol. 14, no. 4, 1996, pp 421-470.

S. H. Clearwater (ed.), *Market-Based Control : A Paradigm for Distributed Resource Allocation*, World Scientific Publishing, River Edge, N.J., 1996.

R. Conway, W. Maxwell, and W. Miller, Theory of Scheduling, Addison-Wesley, Reading, MA, 1967.

D. Coppersmith and P. Raghavan, "Multidimensional On-line Bin Packing: Algorithms and Worst-Case Analysis," *Operations Research Letters*, Vol. 8, February, 1989.

R. Glushko, J. Tenenbaum, and B. Meltzer, "An XML Framework for Agent-based E-commerce," *Communications of the ACM,* vol. 42, no. 3, 1999, pp. 106-114.\

S. Heragu, R. Graves, B. Kim, and A. Onge, "Intelligent Agent Based Framework for Manufacturing Systems Control," *IEEE Transactions on Systems, Man, and Cybernetics*, forthcoming in 2003.

D. S. Hochbaum and D. B. Shmoys, "Using Dual Approixmation Algorithms for Scheduling Problems: Theoretical and Practical Results, Journal of the ACM, Vol. 34, No. 1, January, 1987.

C. Hsu., *Enterprise Integration and Modeling: the Metadatabase Approach*, Kluwer Academic Publishers, Boston, 1996.

C. Hsu, M. Bouziane, L. Rattner and L. Yee, "Information Resources Management in Heterogeneous Distributed Environments: A Metadatabase Approach," *IEEE Transactions on Software Engineering,* vol. 17, no. 6, 1991, pp 604-625.

C. Hsu and S. Pant, *Planning for Electronic Commerce and Enterprises: A Reference Model,* Kluwer Academic Publishers, Boston, 2000.

C. Hsu and C. Carothers, "A Self-Scheduling Model Using Agent-Base, Peer-to-Peer Negotiation, and Open Common Schema," *Proceedings* 17th *International Conference on Production Research,* Blacksburg, VA, August 2003.

M. Nandula and S. P. Dutta, "Performance Evaluation of an Auction-Based Manufacturing System Using Colored Petri Nets," *International Journal of Production Research,* Vol. 38, No. 38, 2000, pp. 2155-2171.

H. Parunak, “Agents in Overalls: Experiences and Issues in the Development and Deployment of Industrial Agent-Based Systems,” ERIM CEC Report, P.O. Box 134001, Ann Arbor, MI 48113-4001, 2001.

V. Prabhu, “Performance of Real-Time Distributed Arrival Time Control in Heterogeneous Manufacturing Systems,” *IIE Transactions*, Vol. 32, No. 4, 2000, pp. 323-331.

T. Sandholm, “Making Markets and Democracy Work: A Story of Incentives and Computing,” *IJCAI-03 Computers and Thought Award Talk Abstract*, 2003.

M. Stonebraker, P. Aoki, A. Pfeffer, A. Sah, J. Sidell, C. Staelin and A. Yu, "Mariposa: A Wide Area Distributed Database System,” *International Journal on Very Large Databases*, Vol. 5, No. 1, 1996, pp. 48—63.

J. Swaminathan, S.F. Smith and N. Sadeh, “Modeling Supply Chain Dynamics: a Multi-Agent Approach”, *Decision Sciences*, Vol. 29, 1998.

L. Tao, "Shifting Paradigms with the Application Service Provider Model," *IEEE Computer,* vol. 34, no. 10, 2001, pp. 32-39.

W. E. Walsh and M. P. Wellman, “A Market Protocol for Decentralized Task Allocation”, Proceedings International Conference on Multi Agent Systems, 1998.