200001439-9

15 MAR 2000

VIRTUAL MARKETPLACE

FIELD OF THE INVENTION

The present invention relates to an electronic trading system, and, more particularly, to an electronic trading system distributed on a network.

BACKGROUND OF THE INVENTION

Although "Electronic Trading Systems" (or "Electronic Stock Exchanges" or "Trade Crossing Networks") are well known in the prior art, these systems suffer from being overly complex, limited to specific, narrow applications, and difficult to manage and use. In general, these systems allow users to electronically enter orders into a 'book,' or a list of all outstanding orders. These orders represent the desire to execute a transaction to buy or sell an item or security within a set of parameters allowable by the administration of the exchange. As such, they are tailored to specific marketplaces, have very specific parameters, limits, and matching rules. The programs, processes and databases associated with each exchange are very much concerned with the specific items being exchanged. For example, consider the difference

between an options exchange and a steel pipe exchange. The options exchange must have parameters, rules and procedures which are specifically relevant to option contracts as well as the associated underlying security. The steel pipe exchange has its own set of rules and parameters which are relevant to that business.

The current state of the art in electronic exchange systems requires a system specifically tailored to the rules and parameters of the items being traded since the knowledge of those rules is important to the matching of the buyer and seller. For example, to match a buyer and seller of an option on a common stock would require the participants to agree on a specific contract (including strike price, underlying security or index, and expiration date), number of contracts, and price per contract. A steel pipe exchange, on the other hand, would require a match on type, shape, quality, weight, and price. If the exchange does the matching, as is the case with prior art electronic exchanges, it must be adapted to each market separately. Moreover, the interest of the exchange, rather than the participants, is the foremost consideration when determining the matching rules. That is, the matching strategies are determined by the exchange based on criteria such as efficiency of matching and maximization of revenue. Accordingly, the customers must adapt to the facilities provided by the exchange and must formulate their trading strategies accordingly, rather than the exchange adapting to the particular needs of each participant. Thus, the need exists for a trading system that avoids these disadvantages and provides flexibility to the traders.

SUMMARY OF THE INVENTION

The present invention provides a method for trading assets in an asset trading system that avoids the limitations of the prior systems and provides other advantages that will be apparent to

one skilled in the art. Broadly speaking, the asset trading system of this invention comprises a virtual marketplace and a plurality of trader client user workstations interconnected over a network in which trading may be conducted without outside intervention. The method comprises the steps of:

creating and transmitting a trader object associated with the asset to be traded from one of the workstations to the virtual marketplace for providing a received trader object, and creating a dispatch object, where the dispatch object is resident at one of the workstations, and being capable of communicating with the received trader object, the trading system being independent of any rules and parameters associated with the asset being traded;

the received trader object creating a contract object, the contract object comprising trading parameters associated with the asset to be traded; and

executing the contract object in response to receipt of a request to execute the contract object by the received trader object; whereby the asset trade is completed between the user workstation and the virtual marketplace.

BRIEF DESCRIPTION OF THE DRAWINGS

To facilitate further discussion of the invention, the following drawings are provided, in which:

FIG. 1 is a schematic view of a computerized asset trading system according to the principles of this invention;

FIG. 2 is a schematic view of a user work station for use in trading on the system;

FIG. 3 is a schematic view of the environment in which the software representatives used in the system operate;

FIG. 4 is a schematic view of the interaction between software representatives;

FIG. 5 is a schematic diagram of the reusable software representative components;

FIG. 6 is a schematic view of the handling of a contract copy by a trader object on the offering side (left) and accepting side (right);

FIG. 7 is a schematic view of a physical implementation of the system;

FIG. 8A is a flow chart showing an example of the logic flow of a dispatch object in the system of the present invention;

FIG. 8B is a flow chart showing an example of the logic flow of a trader object in the system of the present invention;

FIG. 9 is a diagram of some of the contract object methods that can be implemented by the contract objects created by the trading objects;

FIG. 10 is a diagram of some of the notification filter object methods that can be implemented by the notification filter objects created by the trading objects;

FIG. 11 is a schematic view of the logic flow of the system;

FIG. 12 is a diagram of some of the booth object methods that can be implemented by the booth objects; and

FIG. 13 is a diagram of some of the post object methods that can be implemented by the booth objects.

These drawings are for illustrative purposes and should not be used to unduly limit the scope of the invention.

DETAILED DESCRIPTION OF THE INVENTION AND PREFERRED EMBODIMENT

An asset trading system according to the principles of this invention is indicated generally as 20 in Fig. 1. The system 20 is shown connected via a supporting network 22 to a plurality of

user work stations 24. This supporting network 22 may be a public network, for example, the Internet, or a private network. This network is maintained for purposes of accessing the system 20. Preferably, the network 22 is one that implements the TCP/IP protocol. Because trading is conducted on the system 20 without the need for outside instruction or intervention, the network 22 does not have to be a high speed network in order to preserve the fairness of, or equal access to, trading on the system. With the work stations 24, users of the system 20 can create a software representative comprising a trader object 26 that is transmitted to the system 20, and a dispatch object 28 that remains resident on the work station 24 for communicating with the trader object 26.

As shown in Fig. 2, the work stations 24 preferably have a Java enabled web browser 100, such as is available from Netscape Communications, Inc. The browser 100 provides the execution environment for all software except the trader objects which are transmitted to and run in the system 20. Preferably, programs provided by the system are written in the Java language so that they can run on all major computer and operating system platforms. The server also provides the documentation view and authentication/encryption facilities.

The work station 24 also includes authentication certificates 102 for positive identification of an account using a Virtual Marketplace. The authentication certificates and encryption keys are used for authentication and secure transmission of trader objects. They are provided by the system 20 at the time of account set-up. The work station 24 also has a pool of stock software representative 104, as well as a set of software representative authoring tools 106, a software representative component library 108, and a marketplace simulator 110 for testing software representatives. System documentation 112 may also be provided on a server 48 in the system 20, where it can be easily maintained using any of a large number of available web

authoring tools. The documentation can be made up of a series of HTML (hypertext markup language) pages and provides operating instructions for the various software tools, as well as information needed to construct software representatives.

The authentication certificates 102, the software representative pool 104, and the software component library 108 may be stored locally at the work station 24, for example, on a hard drive 114 in the work station. The software representative authoring tools 106, the system simulator 110, the documentation 112, and the representative component library 108 may be stored on the server 48 that is part of the system 20.

The software representative component library 108 is a set of object orientated classes (preferably written in Java) used to support the software representative development. The library 108 includes abstract base classes for the trader object, the dispatch object, the contract object, the notification object, interfaces to the booth object, and support classes for managing money, dates, asset identify, and other commonly used objects. Individual component classes can be loaded from the system on demand.

Fig. 5 shows some of the reusable components offered to make construction of the software representatives easier. They include an abstract trader object class 116, an abstract contract class 118, an asset class 120, a money class 122, an abstract notification filter class 124, an abstract contract list class 126, and an abstract dispatch object class 128. These classes provide general and fundamental functionality of the various object types. They will generally be enhanced by inheritance and customization to implement specific trader strategies. The only classes which are necessary are the abstract trader object class 116, and the abstract contract class 118 or the abstract notification filter class 124. The others are additional, non-essential tools or guideline templates that save development time. Also required from the software representative

component library 108 is a booth interface specification from which the trader objects receive their facilities from the system 20.

The software representative authoring tools 106 are also object orientated classes (preferably written in Java) which can be loaded on demand from the system 20. They are preferably program generators with user-friendly graphical interfaces to provide help in constructing new software representatives.

The market place simulator 110 is a program run from within the browser 100 (preferably a Java applet) which is loaded from the system 20. The simulator 110 allows the user to test software representatives under controlled and repeatable conditions where the user's assets are not at risk. The simulator 110 provides the same booth API (application program interface) as the system 20, so that a software representative that operates satisfactorily on simulator 110 can run unchanged on the system. The simulator 110 takes advantage of independent execution threads available in Java to allow many software representatives to run simultaneously. Both the system 20 and the simulator 110 allow logging of messages by the trader object for debugging and audit trail purposes.

The stock software representatives in the software representative pool 104 are available on the system 20 to be used, tested under simulation, or extended to make new software representatives.

Fig. 6 shows the internal components of a sample trader object 26'. It is a vendor/buyer trader object because its sell strategy is to put contracts for assets to sell in the marketplace and wait until buyers buy them. If the assets are not being bought, the trader object 26' lowers the target price over time until the low price limit is reached. The buyer is passive in the process, just shopping until a sufficient quantity of the assets are found which are priced less than or equal

to its target price. The target price is adjusted downwardly by the seller, and upward by the buyer to the goal limit at a rate determined by the aggressiveness of the goal.

As shown in Fig. 1, the system 20 comprises a plurality of booth processes 30, one for each trader object 26 transmitted to the system. The system 20 also comprises a plurality of post processes 32, one for each asset traded on the system. The booth processes 30 and the post processes 32 are connected by high speed messaging paths 34. As shown in Figure 7, the system 20 is preferably constructed with mirror image or redundant sites at two geographically remote locations, to protect the integrity of the system should there be an interruption at one of the sites.

As shown in Fig. 7, the booth processes 30 can be physically implemented by one or more booth processors 36, which can be any suitable conventional processor, for example a DEC ALPHA processor. Similarly, the post processes 32 can be physically implemented by one or more post processors 38, which can be any suitable conventional processor, for example a DEC ALPHA processor. The paths 34 can be physically implemented by a local area network (LAN) 40 between the booth processors 34 and the post processors 36. The LAN 40 is preferably connected to the supporting network 22 through a router and firewall 42. The routers include booth assignment processors for finding free booths for new trader objects as they enter the system. They can also handle security, and send exception messages to the dispatch objects in the event of a failure of a booth processor. The booths which will receive the trader objects may be distributed across the booth processors 36 across parallel or redundant sites. The posts are also distributed across the sites, but a representation for each asset traded must be made at each site. If a booth processor fails, its trader objects can be terminated and re-introduced by the participants system in response to the loss of connection between the dispatch object and the trader object or the failure of the trader object to check in at a pre-designated time. If a post

processor fails, however, the state of the mark and the contract lists must not be compromised. This may be accomplished by having a post at each redundant site working in tandem with the other. Since trader objects interested in a given post may reside on many processors, the notification method may be by a network multicast message. Copies reach interested booth processes as well as the redundant post processes at the same time. Pickups then, can be directed to the local post processor unless it has failed.

The distribution of contract objects by multicast also allows for a high degree of scalability within the system, because multiple trading objects are performing matching algorithms in parallel within the active booth processes. As is generally the case, intersite multicast messages are delivered later than intrasite messages, and the booth processes can impose a matching delay to reduce unfairness.

The system 20 also includes a clearing and bookkeeping processor 44 on the LAN 40. All significant events including contract registration, pick up, exercise, and cancellation are journalled by the clearing and bookkeeping processors 302. Logging of the encrypted contents of trader objects is also required for an internal audit trail to help resolve potential disputes and to help find logic flaws within the system of the trader objects. Billing for system fees and clearing are also performed by these processors. The clearing processors may be redundant across the sites in the same manner as the post processors.

As shown in Fig. 3, a trader uses a work station 24 to create a software representative comprising a trader object 26 and a dispatch object 28, and sends the trader object to the system 20 via the network 22. The trader object is transmitted to a booth and given goals by its dispatch object 28. For example, as shown in Fig. 6, the goals are price range, quantity range, aggressiveness, buy/sell, and the asset to be traded. The sample trader object is both a buyer and

a seller in one trader object and modifies its behavior based on the buy or sell goal. The trader object 26 is programmed to write contract objects 50. A contract object 50 is derived from an existing contract class or a class derived from a contract class, and represents an option to buy or sell a particular asset. A contract object 50 may contain a start time, a stop time, a pick-up expiration interval, a price, a quantity, and an asset identify. A contract object 50 can contain other information needed by the trader object for its own purposes, for example where the asset being traded is a commodity, grade, style of delivery or location information may be added.

The contract objects 50 can perform the various methods shown in Fig. 9, including a Register Contract method 200, a Registration Complete Handler 202, a Contract Expiration Handler method 204, a Pickup Contract method 206, a Drop Contract method 208, an Exercise Contract method 210, a Contract Exercise Handler method 212, a Cancel Contract method 214, and a Contract Cancellation Handler 216.

The trader objects 26 also create notification filter objects 52. These notification filter objects 52 help the trader objects 26 obtain information about the trading of assets of interest. At least some of the trader objects can invoke the notification object methods shown in Figure 10, including a Register Notification method 210, a Contract Registration Notification Handler 222, a Contract Create Request Handler method 224, and Contract Copy Request Handler method 226, a Cancel Notification method 228, and a Notification Cancellation Handler method 230.

The trader objects 26 maintain several lists for handling the contract objects 50, best shown in Figure 6. On the offering side of a transaction (shown on the left side of Fig. 6), after a trader object 26' creates a contract object 50, the trading object puts a copy of the contract object in its pending registration contract list 54. After the contract is registered at the appropriate post process 32, the registering post invokes the contract's registration complete handler, which

causes the contract to place itself in the registered contract list 56. If another trading object were to subsequently pick up and exercise the contract object, the original contract's exercise handler would be invoked causing the contract to place itself into the exercised contract list 66, as described below. A list management object can be provided on the system 20, which can tally the quantity and average share price of the contracts within, making it easier for the trader object to determine when it has met its goal. The trader object can take advantage of the Check Post method 400 (Fig. 13), which at Step 400a invokes the contract object's contract expiration handler to refine its target price, and re-register the contract with a lower price. The amount and the speed of the reduction is determined by the aggressiveness goal. A more complex trading object might offer some contracts at different prices than others, hoping to sell the full quantity at a higher average price. It also might watch for buy contracts and cancel its own to make a better or quicker trade.

On the accepting side of a transaction (shown on the right side of Fig. 6), when a trader object 26' receives a copy of a contract object 52 pursuant to a notification filter object 54, the trader object puts the copy of the contract object into the filtered contract list 68. If the trader object 26 decides to pick up the contract object 50, the trader object invokes its pick up contract method (described below) and moves the copy of the contract object to the pending pick-up Contract list 60. If the trader object 26 successfully picks up the contract object 50, the trader object is notified and moves the contract to the picked up contract list 62. If the trader object 26 then decides to exercise the contract object 50, the trader object invokes its exercise contract method (described below) and the trader object moves the copy of the contract object on to the pending exercise contract list 64. If the exercise of the contract is successful, then whether on the

buy side or the sell side, the trader object moves the copy of the contract to the exercised contract list 66.

Each booth process 30 includes a booth interface 68 (Fig. 3) for communicating with its respective trader object. Each booth process also includes a statistics keeper 70 (Fig. 3) for maintaining system statistics. Each booth process 30 is capable of invoking a number of methods shown in Figure 12 to facilitate the function of its respective trader object 26. These methods include a Register Contract method 300 a Cancel Contract method 302, a Pick Up Contract method 304, a Drop Picked Up Contract method 306, an Exercise Contract method 308, a Request Post Information method 310, a Log Message method 312, a Register Notification method 314, a Set Limits method 316, a Cancel Notification Method 318, a get Booth Information method 320 and a Exit Handler method 322. The booth processes 30 execute these methods to facilitate the communication of their respective trader objects and the appropriate post processes.

Each post process 32 maintains a list 74 of registered contract objects 50 registered by the various trader objects 26, a list 76 of picked up contracts registered by the various trader objects, and a list 78 of notification filter objects registered by the various trader objects to obtain information about the asset being handled by the post. Each post process includes a post manager object 80 (Fig. 1) for invoking the methods shown in Fig. 13. These include a Check Post method 400, a post Cleanup method 402, a Register Contract method 404, a Cancel Contract or Notification method 406, a Pick Up Contract method 408, a Drop Contract method 410, an Exercise Contract method 412, a Register Notification method 414, and a Request Post Information method 416.

When a trader object 26 creates a contract object 50, it invokes that contract object's Register Contract Method 200 (Fig. 9). The contract object's Register Contract method at step 200a invokes the booth process' Register Contract method 300 (Fig. 12). The booth process' Register Contract method 300 at step 300a finds the proper post process 32, and invokes its Register Contract method 404, and at step 300b puts a copy of the contract on the trader objects pending registration list 54. The post process' Register Contract Method at step 404a puts the contract object in the registered contract list 74, and checks the list 78 of registered filter objects, and sends a copy of the contract object to each matching filter object, and to the originating trading object, which moves the copy of the contract from the pending registration contract list 54 on the registered contract list 56. Further, the contract object's Registration Complete Handler method 202 is invoked by the Booth object after invoking the Post's Register contract method.

As copies of the contract object are sent to the interested trader objects 26, the notification object's Contract Create Request Handler method 224 (Fig. 10) is invoked to allow the receiving trader object to create a copy of the contract within the trader object's environment. The receiving trader object 26 may not use the same contract type and handlers as the original contract. In fact, the original contract object and the notification copy, while derived from the same abstract base class supplied by the system 20, will be derived differently and will contain different functionally. There are some attributes or data items which belong to the base class and will always be present. Those attributes are copied into the new contract when the receiving booth process invokes the notification object's Contract Copy Request Handler method 226 (Fig. 10). This method can also test the contract object being copied to see if there are any other attributes available that are conventional in the market for the asset being traded, and copy them

as well. This feature can be used for many purposes, from passing information between cooperating trading objects and for passing extra terms and conditions suitable to special marketplaces which are invisible to the system. An example might be special settlement and delivery conditions which are presented by convention or are required by a third party post management object. This feature gives the system 20 the flexibility to adapt to many different market place requirements.

Once the contract object has been transformed into the receiving trader object's contract form, its Registration Complete Handler method 202 (Fig. 9) is invoked. Because the copy is now cast into the receiving trading object's custom type and has its custom methods, the registration complete handler performs logic specific to the receiving trading object. In other words, the Registration Complete Handlers of the original contract and of the receiving copy can be completely different. This is desirable because they serve different trader objects, with different strategies. Any events which happen to the contract object registered at the post process can be conveyed to a received copy by invoking the handler methods exercise, cancel, picked-up and expire of the contract copy by the Booth, from the Post. This allows the receiving trader object's contract copies to behave according to the receiving trader object's rules when events happen. Any further filtering of contracts is done by the receiving trader object. This may be necessary because of additional parameters added by convention or required by third party post managers or because market conditions have changed since the registration of the notification.

Notification filter objects 52 are simpler than contract objects 50 because notification objects 52 are only used by the originating trader object 26 and the post manager object 80 of the post process that handles the particular asset. The notification object 52 supplied by the system 20 as part of the component library 108 (Fig. 2) is an abstract base class just like the contract

object. To make the class useful it must be extended through object inheritance to include functions and handlers specific to the originating class (a class is a template for one or more objects), it invokes the notification object's Register Notification method 220 (Fig. 10). The notification object's Register Notification method 220 at step 220a invokes the booth process' Register Notification method 314 (Fig. 12). The booth process' Register Notification method at step 314a finds the proper post process 32 and invokes that post process' Register Notification method 414 (Fig. 13). The post process' Register Notification method at step 414a puts the notification object in the post process' notification list 78, and at step 414b checks all contract objects in the registered contract list 74 to produce a series of contract notifications for the newly registered filter. This allows trader objects 26 which enter the system late to be as well informed about the market as trader objects which have been there for a longer time, and to begin work immediately with no start-up delay. Notification objects 52 are given several parameters when they are created; the item of interest (this identifies the post process), the price range of interest, the quantity range of interest, whether to be notified of buy or sell contracts, and the acceptable expiration timer limits.

When a registered object is picked up by a receiving trader object, it is moved from the registered contract list 74 (Fig. 3) to the picked up contract list 76 (Fig. 3).

A background process 500 (Fig. 11) at step 506 repeatedly invokes each post process' Check Post method 400 (Fig. 13). The post processes' Check Post process 400 at step 400a checks the post process' registered contract list 74 and invokes each expired contract object's Contract Expiration Handler method 204 (Fig. 9) both for the original contract object and all of its copies. The contract object's Contract Expiration Handler method at step 204a tests the trader object's exercised contract list 66 to determine whether the quantity goal has been reached. If the

goal has been reached, at step 204b the contract object is removed from the registered contract list 56. If the goal has not been reached, at step 204c a change in strategy, for example, a target price adjustment is made, and at step 204d the revised contract object is registered. The post process' Check Post method 400 also checks the picked up contract list 76 (Fig. 3) for contract objects which have passed their pick up expiration interval, and moves them back to the Registered Contract list 74, and their copies are notified by invoking their Contract Expiration Handler method 204 (Fig. 9).

If a trading object 26 decides to cancel a contract object 50, the trading object can invoke the contract object's Cancel Contract method 214 (Fig. 9). The contract object's Cancel Contract method 214, at step 214a, invokes the booth process' Cancel Contract method 302 (Fig. 12). The booth process' Cancel Contract method 302, at step 302a, finds the appropriate post process 32, and invokes the process's Cancel Contract method 406. The post process' Cancel Contract method 406, at step 406a, removes the contract object from the registered contract list 74.

Similarly, if a trader object decides to cancel a notification object 52, the trader object can invoke the notification object's Cancel Notification method 228 (Fig. 10). The notification object's Cancel Notification method 228, at step 228a, invokes the booth process' Cancel Notification method 318 (Fig. 12). The booth process' Cancel Notification method 318, at step 318a, finds the appropriate post process 32, and invokes the post process's Cancel Notification method 406. The post process' Cancel Notification method 406, at step 406a, removes the notification object from the notification list 78.

If a trader object 26 wishes to drop a picked-up contract before its expiration, the trading object can invoke the contract object's Drop Contract method 208 (Fig. 9). The contract object's Drop Contract method 208, at step 208a, invokes the booth process' Drop Picked Up Contract

method 306. The booth process' Drop Picked Up Contract method 306, at step 306a, finds the proper post process 32, and invokes the post process' Drop Contract method 410. The post process' Drop Contract method 410, at step 410a, removes the contract from the Picked Up Contract list 76, and re-notifies the contract

In operation, the system 20 simply facilitates operation of the various trader objects submitted by users. This is illustrated in Figure 4, where a first trader object 26A in booth process 30A, programmed by its user to sell asset C, creates a contract object 50A for the sale of asset C, specifying quantity, price, and other relevant transaction parameters. The trader object 26A then registers the contract object by invoking the contract object's Register Contract method 200 (Fig. 9). This method includes the step 200a of invoking the booth's Contract Registration method 300, and the step 200b of putting the contract object on the trading object's pending registration contract list 54 (Fig. 6). In response, the booth 30A invokes its Register Contract method 300 (Fig. 12). This method includes the step 300a of finding the proper post process 32 for the particular asset (in this case post process 32C) and invoking that post process' Register Contract method 404. In response, the appropriate post process 34C invokes its Register Contract method 404 (Fig. 13). This method includes the step 404a of putting the contract object 50A into the Registered Contract List 74 (Figs. 3 and 4), and the step 404b of checking all notification filter objects in the notification list 78 (Fig. 3 and 4), and sending a copy of the contract object to each matching notification filter 52. Upon completion of the notification, the trader object 26A is notified by the registration complete handle 202, which is invoked by the booth, upon notification from an appropriate post and invokes the contract object's Registration Complete Handler method 202 (Fig. 9). This method 202 includes the step 202a of moving the

contract object from the trading object's pending registration contract list 54 (Fig. 6) to the registered contract list 56 (Fig. 6).

If the trader object 26A wishes to revoke the contract object 50A, it invokes its Cancel Contract method 214 (Fig. 9). This method 214 includes the step 214a of invoking the booth process' Cancel Contract method 302. In response, the booth process 30A invokes its Cancel Contract method 302 (Fig. 12). This method includes step 302a of finding the proper post process 32C and invoking that post process' Contract Cancellation method 406. In response, the post process 32C invokes its Cancel Contract or Notification method 406. This method includes the step of 406a of removing the contract object 50A from the post process' registered contract list 74, and calling the cancel handle, as detailed above.

Meanwhile, another trader object 26B in booth process 30B, programmed by its user to purchase asset C, creates a notification filter object 52B with quantity, price, and other parameters that will cause contract objects of potential interest to be communicated to the trader object 26B. The trader object 26B invokes the notification filter object's Register Notification method 220 (Fig. 10). This method 220 includes the step 220a of invoking the booth process's Register Notification method 314. In response, the booth 26B invokes its Register Notification method 314 (Fig. 12), which includes the step 314a of finding the proper post process 32C for the particular asset, and invoking that post process's Register Notification method 414. In response, the appropriate post process 32C invokes its Register Notification method 414 (Fig. 13). This method includes step 414a of putting the notification filter object 44B into the post process' notification list 78 (Fig. 3 and 4), and the step 414b of checking all the registered contracts list 74 (Fig. 3 and 4) and sending a copy of each matching contract object to the trader object 26B.

If the trader object 26B wishes to remove its notification filter object 52B, the trader object invokes the notification filter object's Cancel Notification method 228 (Fig. 10). This method 228 includes the step 228a of invoking the booth process' Cancel Notification method 318 (Fig. 12). In response the booth process invokes its Cancel Notification method 318. This method includes the step 318a of finding the proper post process 32C and invoking that post process' Cancel Notification method 406. This method includes the step of 406a of removing the notification object 52b from the post process' notification list 78.

The trader object 26B receives copies of contract objects 50 pursuant to its notification request filter object 52B, and the trader object 26B invokes the notification filter object's Contract create Request handler method 224 (Fig. 10). This method 224 includes the step 224a of creating a new copy of the contract object of the proper type of Notification Copy. The receiving trading object 26B then invokes its Contract Copy Request Handler method 226 (Fig. 10). This method 226 includes the step 226a of copying important contract attributes from the replicated original to the local notification copy created during method 224. Once the contact object has been transformed into the receiving trading object's contact form, the contract object's Registration Complete Handler method 202 (Fig. 9) is invoked. Because the copy is now cast into the receiving trading object's custom type, and has its custom methods, the Registration Complete Handler method performs logic specific to the receiving trading object 26B. That is, the Registration Complete Handler methods of the original contract object and the receiving copy can be completely different. This is desirable because these contract objects serve different trading objects with different trading strategies. Any events which affect the contract registered at the post can be conveyed to the received copy by invoking the handler methods as discussed

above. This allows the receiving trader object's contract copies to behave according to the receiving trader object's rules when events happen.

If the contract object 50A is acceptable to trader object 26B, then the trader object 26B invokes its Pickup Contract method 206 (Fig. 9). This method 206 includes the step 206a of moving the contract object 50A to the pending pickup contact list 60 (Fig. 6), and the step 206b of invoking the booth process' Pickup Contract method 304. In response, the booth process 26B invokes its Pickup Contract method 304 (figure 12). This method 304 includes the step 304a of finding the proper post process, and invoking the post process' Pickup Contract method 408 (Fig. 13). In response, post process 34C invokes its Pickup Contract method 408 (Fig. 13). This method 408 includes the step 408a of determining whether the contract object is a copy and registered, and if so moving the copy to the post process' picked up contract list 76 (Figs. 3 and 4); and step 408b of concealing all other copies of the contract by invoking the Contract Picked Up Handler method 209 of the copy of the contract object. This method includes the step of moving the contract object from the pending picked up contract list 60 to the picked up contract list 62.

If the trader object 26B subsequently decides to exercise the contract, the trader object invokes its Exercise Contract method 210 (Fig. 9). This method 210 includes the step 210a of moving the contract to the Pending Exercise list 64 (Fig. 6), and the step 210b of invoking the booth process' Exercise Contract method 308 (Fig. 12). This method 312 includes the step 308a of finding the proper post process and invoking that process' Exercise Contract method 412 (Fig. 13). In response, the post process 32C invokes its Exercise Contract method 412 (Fig. 13). This method 412 includes the step 412a of determining whether the contract object was properly picked up by the exercised Trader Object, and if so removing the contract object from the picked

up contract list 76 (Fig. 3); the step 412b of invoking the exercised handler method 212 on the original contract and the exercising copy; and the step 412c of updating the post statistics 70. In response the original contract and the exercising copy invoke their Contract Exercise handler method 212 (Fig. 9). This method 212 includes the step 212a of moving the contract to the exercised list 66.

The system provides fairness to the trader objects 26 operating in the system in two ways: (1) allowing all trader objects the same amount of processor time; and (2) randomizing the pickup process. The equal allocation of processing time can be easily accomplished by a timesliced processor schedule found in many operating systems. The randomization of the pickup process is important so that any given trading object does not consistently win the pickup race more often than the others. It is not practical to simultaneously notify every registered trading object on a list of registered trading objects, but when notification is sent to the listed registered trader objects, it must not consistently arrive at one trader object sooner than the others.

An example of the logic flow of a software representative created on a work station for use with the system 20 is shown in Figs. 8A and 8B. As shown in Fig. 8A the dispatch object 28 logic might follow the following steps: The process begins at step 600. At step 602, the dispatch object inquires whether the user wants to buy or sell, the identity of the asset to be traded, the quantity, the starting price, the price limit, and the time limit. At step 604 the dispatch object transmits the trader object to the system 20. At step 606, the dispatch object 28 invokes the trader object's run method, and passes the user's goals as parameters. At step 608, the dispatch object can remain active to catch any exceptions not handled by the trader object, although it is anticipated that the trader objects can operate completely independently. Independent operation

eliminates a user's objectives from being jeopardized from any discrepancies in access to the network 22.

As shown in Figure 8B, the trader object logic might follow the following steps. At 700, the dispatch object 28 invokes the trader object's run method (step 606 in Fig. 8A). At step 702, the trader object 26 decides whether the goal is to buy or sell. If the goal is not to sell, then the logic proceeds to step 704. If the goal is to sell, then the logic proceeds to step 706. At step 706 the trader object sets the target price at the upper limit. At step 708 the trader object divides the quantity into marketable sizes. At step 710, the trader object 26 creates short term contract objects with the appropriate price and quantity parameters. At step 712 the trader object registers the contract objects created. At step 714 the trader object 26 continues to monitor whether the exercised list 66 contains a quantity within the quantity goals. If so, the logic proceeds to step 716 where the trader object returns the results to the dispatch object 28, and exits. This, of course, is only one example of the possible logic for a passive-selling trader object.

If the goal at step 702 was to buy, then the logic proceeds from step 704. At step 704 the trader object sets the target price at the lower limit. At step 718 the trader object creates a notification filter object with limits matching the goals. At step 720, the trader object registers the notification filter object. At step 722 the trader object determine whether the contracts in the notified list represent a quantity that is greater than or equal to the minimum goal. If not, the logic continues to cycle until the contract objects representing the minimum quantity goal are available. If the contracts meet the minimum quantity goal, then at step 724 the trader objects picks up a set of contracts with the lowest cost for a quantity between the quantity limits. At step 726 the trader object moves the contracts to the Pending Picked Up Contract list 60. At step 728 the trader object waits until all of the contracts have been successfully picked up. At step 730 the

trader object determines whether the average price is below the maximum goal. If it is not, then the trader object returns to step 722. If the average cost is below the target, the trader object moves to step 732, moving the contracts to the Pending Exercise List 64, and at step 734 exercises the contracts in the picked up list. At step 736 the trader objects waits until all of the contracts have been exercised. At step 738 the trader objects reports the results to the user, and exits.

At the initialization of the system 20, the system runs the background method 500 (Fig. 11). At step 502 the system creates a post process 32 for each type of asset to be traded on the system. At step 504 the system 20 creates a booth process 30 for the maximum number of simultaneous trader objects that will be permitted on the system. At step 506 the system 20 continually cycles to invoke the Check Post method of each of the post process 32.

When a trader object 26 is delivered to the system 20, the system implements the load booth method 508. At step 510 of the method 508, the system receives, and authenticates (and possibly decrypts) trader object 26. At step 512 the system creates an instance of the specified trader object 26. At step 514 the system invokes the trader object's run method and passes parameters received from its dispatch object 2.

The system allows the posts to be individually managed. In the case of a managed post, the system 20 implements the load post method 516. At step 518 the system creates an instance of a specified post manager object 80. At step 520 the system 20 invokes the post manager object run method and passes parameters received from its dispatch object.

Variations and modifications will be apparent to those skilled in the art, and the claims are intended to cover all variations and modifications that fall within the true spirit and scope of the invention.

WHAT IS CLAIMED IS:

1. A method for trading assets in an asset trading system, said trading system comprising a virtual marketplace and a plurality of trader client user workstations interconnected over a network in which trading may be conducted without outside intervention, said method comprising the steps of:

creating and transmitting a trader object associated with said asset to be traded from one of said workstations to said virtual marketplace for providing a received trader object, and creating a dispatch object, said dispatch object being resident at said one workstation, and being capable of communicating with said received trader object, said trading system being independent of any rules and parameters associated with said asset being traded;

said received trader object creating a contract object, said contract object comprising trading parameters associated with said asset to be traded; and

executing said contract object in response to receipt of a request to execute said contract object by said received trader object; whereby said asset trade is completed between said user workstation and said virtual marketplace.

2. The method of claim 1, wherein said network is a public Internet.

3. The method of claim 1, wherein said network is a private network.

4. The method of claim 1, wherein said step of creating and transmitting said trader object further comprises the steps of:

creating a software representative, said software representative comprising said trader object and said dispatch object, said software representative residing on said trader client user workstation, said software representative transmitting said trader object to said virtual marketplace.

5. The method of claim 1, wherein said the step of creating said trader object comprises the step of creating a notification object, said notification object transmitting an exercise contract request to a second contract object.

6. The method of claim 1, wherein said contract object comprises an asset identifier, a price identifier, a quantity identifier, and a time identifier.

7. The method of claim 6, wherein said contract object changes the value of said price identifier when said time identifier has expired.

8. The method of claim 1, wherein said trader object notifies said dispatch object when said contract has been exercised.

9. The method of claim 5, wherein said notification object notifies said dispatch object when said second contract has been exercised.

10. The method of claim 1, wherein said virtual marketplace comprises a plurality of booths and a plurality of post processes, wherein said step of transmitting said trader object to said virtual marketplace further comprises the step of transmitting said trader object to one of said

* C L M S *

booths, wherein said one of said booths registers said asset associated with said contract object in one of said post processes.

11. The method of claim 10, wherein a second trader object and said trader object agree to said trading parameters associated with said asset to be traded, and wherein said one of said post processes notifies a second dispatch object associated with said second trader object that said trading parameters associated with said asset to be traded have been agreed to.

12. The method of claim 10, wherein a second trader object and said trader object agree to said trading parameters associated with said asset to be traded, and wherein said one of said post processes notifies said dispatch object associated with said trader object that said trading parameters associated with said asset to be traded have been agreed to.

