December 2003

The Design of Intelligent Agent Supported Exception Management in Securities Trading

Minhong Wang
City University of Hong Kong

Huaiqing Wang
City University of Hong Kong

Kowk-Kit Wan
City University of Hong Kong

Dongming Xu
City University of Hong Kong

The Design of Intelligent Agent Supported Exception Management in Securities Trading[1]

Minhong Wang
City University of Hong Kong
iswmh@islcityu.edu.hk

Huaiqing Wang
City University of Hong Kong
iswang@is.cityu.edu.hk

Kwok Kit Wan
City University of Hong Kong
iskk@is.cityu.edu.hk

Dongming Xu
City University of Hong Kong
isxu@is.cityu.edu.hk

Abstract

With rising trading volumes in securities transactions and increasing risks faced by global financial firms and markets, STP (Straight Through Processing) is internationally recognized by the securities industry, which is to manage securities trades throughout the trade lifecycle automatically and without human intervention. However, while attempting to achieve STP, care must be taken to ensure that trades giving rise to risk, or trades containing errors, are detected and reconciled at the earliest opportunity. In this paper we focus on the exception management in securities transactions, which attempts to pass the trade information within the trade lifecycle in a timely and accurately fashion. In our system, various classes of intelligent agents are proposed for exception management in securities trading. Monitoring agents keep track on the trading processes; a diagnostic agent is to identify the problems when monitoring agents find exceptions; a resolution agent will take some initiatives to resolve problems based on the output from the diagnostic agent. Collaboration within such agents will enhance the ability of exception management for STP.

Keywords: Intelligent agent, exception management, securities trade, straight through processing

Introduction

In today's fast moving market the financial institutions are facing what many consider to be a great challenge - the transition to Straight Through Processing (STP) to shorten the settlement cycle of securities trades. STP is the end-to-end automation of securities trading process from order to settlement (Securities Industry Association 2002). It is not just the processing of fully automated trade and settlement cycle, but also the automation of all processes required for identifying and fixing any exceptions reported in the cycle. This requires participants to enable an exception-based process to achieve some level of straight through processing, which embodies early identification of errors for timely resolution (Simmons 2001). However, the major issue of exception management has not been adequately addressed in STP initiatives. The mechanism of exception management in securities transactions is still under investigation.

In this paper we will focus on the exception management in securities transactions, which attempts to make sure the information passed to all parties involved in the transaction process are up-to-date, complete and reflect reality. In order to provide a novel solution of exception management in securities trading, we propose to apply the technology of intelligent agent to deal with the representative abnormal transactions within the trade lifecycle. Some properties of intelligent agents, such as reactive and

[1]This research is supported by a Strategic Research Grant (7001309) from City University of Hong Kong.

proactive behaviours, are directly applicable to tackling abnormal transactions in a systematic and goal oriented manner. In our system, a set of intelligent agents is applied to the STP environment. Several monitoring agents keep track on the trading processes; when monitoring agents find exceptions, a diagnostic agent will be initiated and attempt to identify the problems; based on the output from the diagnostic agent, a resolution agent will take some initiatives to resolve problems. Such agents may work autonomously and collaboratively to provide some automation and intelligence to exception management in securities trading.

The rest of the paper is organized as the follows. Section 2 briefly reviews the relevant literatures on securities trading and exception management. Section 3 presents a multi-agent system framework for exception management and describes the agent architecture with their communication. In order to demonstrate how these agents work together to reach their goals, the operation process of exception management in our system is described through an example in section 4. Some related work about STP oriented system with exception resolution is introduced in section 5. Finally, section 6 addresses some conclusions as well as the future work.

Securities Trading and Exception Management

With rising trading volumes in domestic and cross-border securities transactions, risks faced by global financial firms and markets are increasing. Significant inefficiencies in cross-border transactions caused by data re-entry, lack of standards, delays, frequent manual intervention, and other breaks in the securities transaction workflow have led to high failure rates, high costs and operational risks in such trades. While every industry undergoes an occasional major change to satisfy customer and market demands, the financial services industry is presently in the throes of STP in order to achieve the possibility of processing trades in a most proficient manner. STP provides a nonstop flow of information from trade execution to settlement; it involves the seamless, automated electronic transfer of trade information to all parties in as close to real-time as possible. However, while attempting to achieve full automation of trade throughput, care must be taken to ensure that exceptional trades giving rise to risk, or trades containing errors, are detected and reconciled at the earliest opportunity. STP is only achievable if the trade information is passed within the trade lifecycle in a timely and accurate fashion.

In recent years, a number of players in the securities markets have implemented STP systems and delivered risk management and competitive advantages as they seek to achieve T+1 settlement, i.e. settling securities trades in one day. However, the major issue of risk management has not been adequately addressed, and the mechanism of exception management is still under investigation. "The market gets confused when they talk about exception management because they are usually talking about people being involved in the resolution (Guerra 2002)." Some exception management systems simply takes a faulty trade which has been kicked out of the STP flow and route it to an operations professional for examination and repair. Some of the more sophisticated products on the market will have a list of remedies which correspond to each of the likely failures. "As those things get automated and more reliable, exception resolution will more quickly move from humans to software (Guerra 2002)." More efforts are required to improve the exception management in securities transactions to support STP environment. Some heuristic techniques are needed for monitoring operations to predict points of failure.

Multi-agent Framework for Exception Management

The concept of intelligent agent has rapidly become an important area of research. Intelligent agents can be seen as software agents that enjoy such properties as autonomy, co-operativity, reactivity, pro-activity and mobility (Wooldridge and Jennings 1995). Actually, an agent is a computer system that is situated in some environment, and that is capable of autonomous action in order to meet its design objectives (Jennings and Wooldridge 1998). A generic agent has a set of goals (intensions), certain capabilities to perform actions, and some knowledge (or beliefs) about its environment. To achieve its goals, an agent needs to reason about its environment (as well as behaviors of other agents), to generate plans and to execute these plans. Multi-agent systems consist of a group of agents, interacting with one another to collectively achieve their goals. By drawing on other agents' knowledge and capabilities, agents can overcome their inherent bounds of intelligence. In recent years, there has been considerable growth of interest in the design of a distributed, intelligent society of agents capable of dealing with complex problems and vast amounts of information collaboratively. The researchers have proposed to design and develop numerous intelligent agent-based systems to support business processes in dynamic and unpredictable environment. It is also proposed to use intelligent agents for business data monitoring, in which intelligent agents can intermediate on behalf of business analysts by being able to perform limitless, error-free routine calculations and interpretation rapidly to the precise requirements of business managers (Wang et al. 2002, Wang and Wang 1997, Barr and Baker 1995).

Based on our past research and experimental results, some properties of intelligent agents, such as reactive and proactive behaviors, are directly applicable to tackling abnormal transactions in a systematic and goal oriented manner (Wang et al. 2002, Wang and Wang 2002, 1997). The stage of automated exception management usually consists of several key stages. Firstly, the relevant data are collected for the observation on securities transactions. Secondly, problem or outstanding transactions are highlighted as possible exceptions. Identifying exceptions as soon as possible after a transaction has been set in motion will deliver considerable benefits. Thirdly, the nature of the problems is identified to determine exactly what is wrong with the individual transaction and whether it can be automatically mitigated. Fourthly, actions such as automated messaging take place in an attempt to repair the problem. Thus, the exception resolution process is decomposed into autonomous tasks, and multi-agent technology can be applied to realize exception management. Each agent can be delegated a particular task to exhibit its goal-oriented and reactive behavior, and cooperate with others to pursue their goals.

Some Issues of Exception Management

An exception is anything that prevents the successful completion of normal business processes. Exception management, at its most basic level, is about repairing what is broken. An exception-management system is one that can track the predictable events of processing a trade throughout its lifecycle, and only when a system knows what is supposed to happen to a trade can it identify errors. Here, we first present the general lifecycle of securities trades in Figure 1. Usually, the trade lifecycle contain order placement, trade execution, trade agreement, trade clearance, settlement instruction, and payment and delivery. The description of each step in the trade lifecycle is provided in Table 1. In order for understanding, the description is from the perspective of the securities trading organizations (STO), which is a collective term that describes those who reside within the securities marketplace, namely traders and market makers, who sell or buy securities from investors, agents or other STOs. After some investigations on possible problems in securities transactions, we find there are higher rates of trade failures related to trade details, trade agreement, and settlement instruction (SI) matching. In another word, there are more possible failures occurring around the three points in the trade lifecycle that have been highlighted in Figure 1. Accordingly, we initiate our exception management solution from the monitoring activities on the three points; the details are illustrated below.

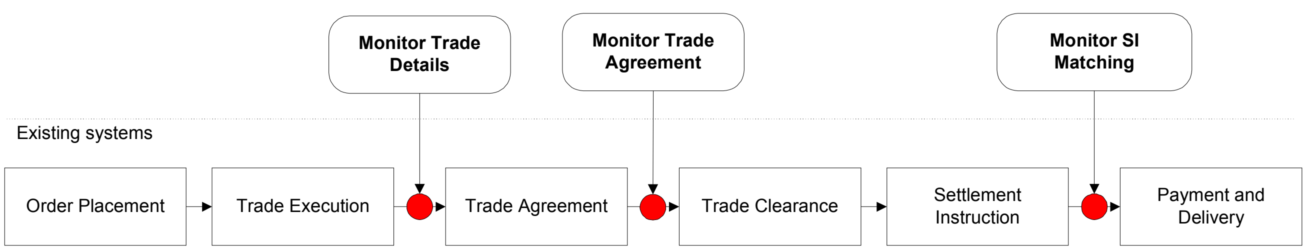

Figure 1. Trade Lifecycle with Some Monitoring Points

Table 1. General Processes in Trade Lifecycle

Order Placement	The investor places an order via the broker to buy or sell some specific securities.
Trade Execution	The trader or market maker executes the order.
Trade Agreement	The trade parties review and confirm the trade details.
Trade Clearance	The clearing company defines net settlement obligation and assigns responsibility for effecting settlement.
Settlement Instruction	The trade parties send settlement instructions (SIs) to their custodians.
Payment & Delivery	The trade is settled by exchanging securities and cashes between the trade parties.

Monitor Trade Details

This is to perform a final check of data contained within a fully enriched trade, in order to reduce the possibility of erroneous information being sent to the outside world. Some risks can arise as a result of trading error, trade recording error and trade enrichment error. For instance, the trader deals a trade at a price that is significantly different from the market price, i.e. a price falling outside the tolerance relative to the market price (e.g. 10% above or below). Or the trade has been captured with some unusual components, such as value date on a public holiday, incorrect calculation of trade cash value, missing custodian details, and so on. The consequence of failing to identify, investigate and resolve such trades early in the trade lifecycle may mean that resolution may well take considerably longer and the risks increase. Though many securities trade organizations have adopted this kind of trade validation, this activity is usually effected manually. It is suggested that performing extensive trade validation is only achievable by use of intelligent and efficient systems (Simmons 2001). In our system a *trade details monitoring agent* is proposed to detect such errors on a trade-by-trade base.

Monitor Trade Agreement

This is to gain agreement of the trade details between the parties to a trade as soon as possible after trade execution. In modern settlement systems, the issuance of trade confirmation can usually be automated. However, this does not necessarily result in the issuer (e.g. STO) being certain that the trade has been agreed with the counterparty. On the one hand, the issuer of the trade confirmation hopes that the recipient will check the detail upon receipt; however it is not always the case. Sometimes the counterparty can not recognize the trade, or did not act upon matching advice in a reasonable timeframe, or failed to receive the confirmation, or failed to transmit the response to the issuer by a transmission error (GSCS Benchmarks 2002). On the other hand, the sheer volume of trading usually precludes STOs from positively investigating whether the counterparty to each trade agrees or disagrees with the trade details. The risk of financial loss cannot be eliminated on trades where it is unknown whether the trade parties agree the trade or trade details. Based on this concern a *trade status monitoring agent* is proposed in our system to keep track on trade agreement and highlight those un-agreed trades for further investigation and resolution.

Monitor SI Matching

The matching of seller's and buyer's trade details is, in many cases, effected through two routes, namely trade agreement and SI matching. These two exercises are similar, but typically the time may be different. Trade agreement is necessary immediately after trade execution, and SI matching is effected between trade execution and value date. Settlement instructions may be alleged by a counterparty that does not subscribe to a trade agreement mechanism. Similarly, we will charge the *trade status monitoring agent* to detect un-matched settlement instructions, such as mismatched instructions, outstanding instructions, instructions issued but not received by a transmission error.

Multi-Agent System Architecture

In this research we will incorporate intelligent agents into STP environment to provide automation and intelligence in exception resolution capability. A multi-agent based exception management system will be developed to link with all the applications through which a trade would pass during its lifecycle. Unlike those commercial STP oriented products, we are not trying to create a new settlement system or reengineer existing systems to deliver exception resolution. Instead, we focus on making use of existing securities trade applications and try to fundamentally use internal resources to build software capabilities to interact with existing systems.

As outlined in Figure 2, several classes of intelligent agents are applied in our system to provide a set of functionalities for exception management in securities trading. *Data acquisition agents* are responsible for collecting trade data from settlement related systems and pass them to monitoring agents. Two kinds of monitoring agents including *trade details monitoring agent* and *trade status monitoring agent* will monitor the trading processes and keep track on exceptions. When monitoring agents find exceptions, a *diagnostic agent* will be initiated to identify the nature of problems. Based on the output from the diagnostic agent, a *resolution agent* will take some initiatives to attempt to resolve problems. The details of such agents are described below.

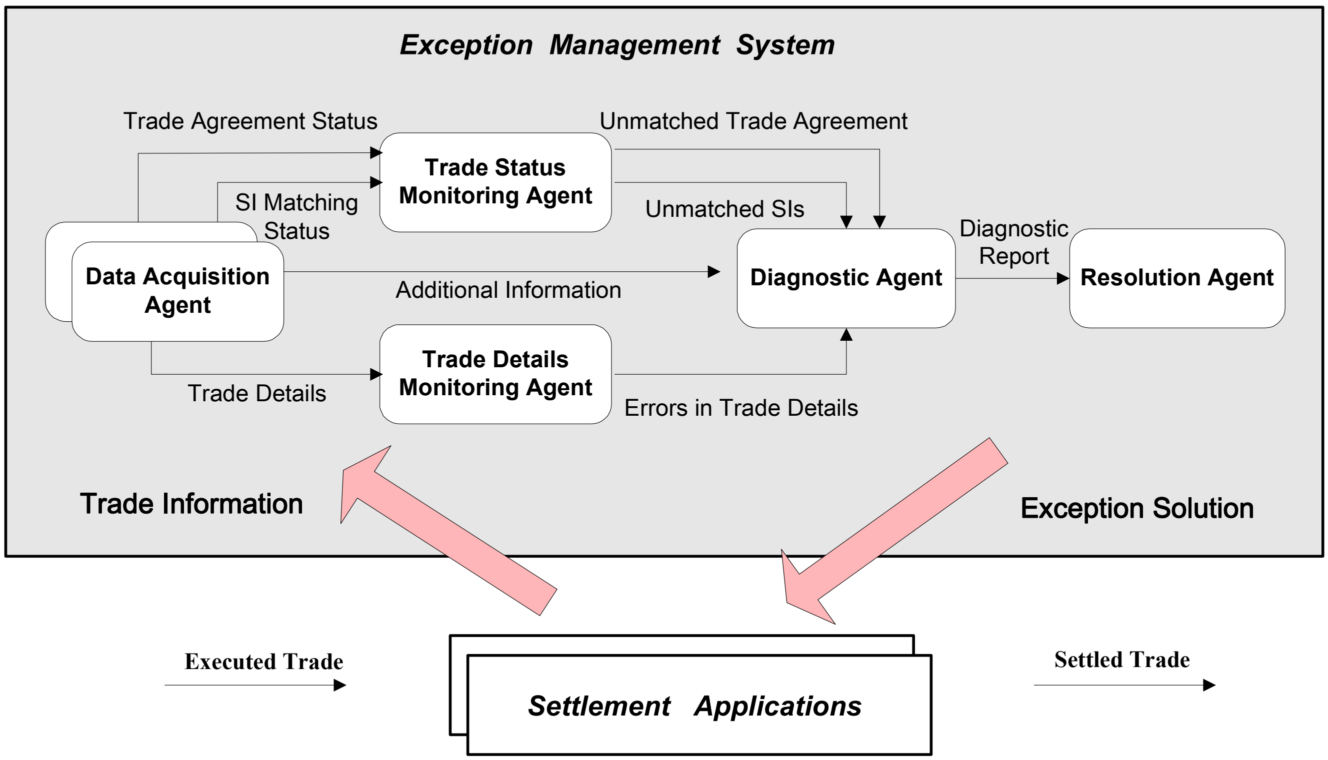

Figure 2. Multi-Agent Based System for Exception Management

The **data acquisition agents** in our system are charged to capture real-time trade data from existing settlement system for the track on securities transactions. Several kinds of trade data such as trade details, trade agreement status, SI matching status, and so on are required to collect for monitoring activities. Based on request from other agent (e.g. the diagnostic agent), the data acquisition agent may search other additional data to support the investigation of problems.

Two kinds of monitoring agent including trade details monitoring agent and trade status monitoring agent are proposed to monitor securities transactions on a trade-by-trade basis. They work autonomously to execute real-time monitoring activities around the three points in the trade lifecycle. Followed is the detail of the two kinds of monitoring agent.

The **trade details monitoring agent** is to detect any error contained within the details of each trade. If an unusual component in a trade is captured, e.g. value date on a public holiday, undistinguished counterparty, irrational quantity, the trade will be sent to the diagnostic agent for further processing.

The **trade status monitoring agent** is applied to keep detection on trade agreement and settlement instruction processes. Those un-agreed trades such as disagreed confirmation, outstanding confirmation, and denied confirmation will be transmitted to the diagnostic agent for further investigation. Similarly, those un-matched settlement instructions such as mismatched SI, outstanding SI, denied SI, and SI advisory will be reported for further investigation and resolution. The SI advisory may take place in the situation that only the one side of trade partners has issued a SI while no SI issued from the counterparty.

The **diagnostic agent** is usually in the waiting mode. Upon receiving the output from monitoring agents, it starts its diagnosing process to investigate the nature of such problems. This agent will conduct analysis on exception reports from monitoring agents as well as other related information to create diagnostic reports. For example, while receiving an exception report about a trade agreement detected in an “outstanding” status, the diagnostic agent may check the transmission history of the trade confirmation to investigate if this trade confirmation has been sent to the counterparty successfully.

Based on the report from the diagnostic agent a **resolution agent** will take some initiatives to resolve problems. Followed the above example, when a trade confirmation is examined not being sent to the counterparty successfully by a transmission error. The resolution agent will ask the settlement system to issue the trade agreement again. If the issuing is failed again, an instant message with transmission error report will be sent to a manager for further processing. Of course, we can only automate the solutions for those exceptions that occur with a high degree of frequency. While automation is key to "de-risking" the process, skilled human input still remains vital. In case where human intervention is required, the resolution agent will send a message to an operation-specialist's desktop.

Agent Architecture

Generally, the design of agent architecture concerns an agent's external interface, operational facility and knowledge base. The external interface envelops an agent and provides access to it via a well-defined interface, and it is also the primary conduit for communication between agents. The operational facility can execute different functions and provide collaboration with other agents. Knowledge is required by each agent to perform its internal and external activities. It consists of resource status information, rules for particular tasks, information about other agents, and so on. In a multi-agent system, software agents are proposed to perform some tasks autonomously on the user's behalf, and this autonomy relies on agents' knowledge about their environment. Knowledge base refers to the knowledge or data which is the agent's perception or awareness of its environment. Various kinds of intelligence are supported by this kind of data (Caglayan and Harrison 1997). The knowledge of other agents, including real trade information, monitoring policies, diagnostic rules, trades in problems, etc., will assist knowledge level solutions to exceptions reconciliation in securities transactions. Business rules extracted from business practice will form an important part of knowledge base of these software agents.

Agent Communication

Interaction among agents, an important aspect on research of multi-agent system, is set up on lower-level data communication as well as control information with semantic and knowledge. The most popular language for agent communication is Knowledge Query and Manipulation Language (KQML) (Finin et al. 1994). One problem in most previous agent systems employing KQML is that there are in fact several different types of KQML (Sycara et al. 1996, Fox et al. 2000). Each system uses its own KQML for its internal use only, and thus the meaning of the language is differed depending on the context (Labrou et al. 1999). Recently, there are some researches focusing on the use of XML (Extensible Markup Language) in agent communication (Glushko et al. 1999). XML is a meta-language, that is, a language used to describe a language. It enables the definition of customized markup languages for different classes of documents. GSTP (Global Straight through Processing) requires using common standards to pass transaction information seamlessly through automated systems, and standards convergence is now taking place with the key trend toward XML (Kumar and David 2000).

Since our exception management system is to work in conjunction with other STP-related applications, open standards like XML will be used for agent communication. We will design our agent communication language based on XML, KQML and STPML. STPML is the Straight Through Processing Markup Language. It is an XML message specification designed for the financial securities trading industry to meet the requirements of straight through processing (Financial Models Company 2003). In our system, agents will send and receive information through XML encoded message with KQML like format. Part of STPML is to be adopted as the information model to represent securities transaction data. In this way, the use of XML, KQML and STPML schema enable software agents to understand the contents of messages correctly and consistently.

An Example

As noted above, all these agents work autonomously and collaboratively in the multi-agent environment. Each agent focuses on its particular task (e.g., data acquisition, monitoring, diagnosing, resolution) without interventions from outside. When a monitoring agent captures a possible exception, the exception report will be issued to the diagnostic agent for further investigation and then sent to the resolution agent for reconciliation. By drawing on other agents' knowledge and capabilities, agents can overcome their inherent bounds of intelligence and work collaboratively to pursue their goals. Although there is no need for centralized storage of all knowledge regarding exception management, there could be one consistent knowledge repository that maintains and integrates all information related to the monitoring and analysis tasks. In this way, the various agents that make up the system can exchange knowledge regarding entities involved and deal with exceptions in a collaborative manner.

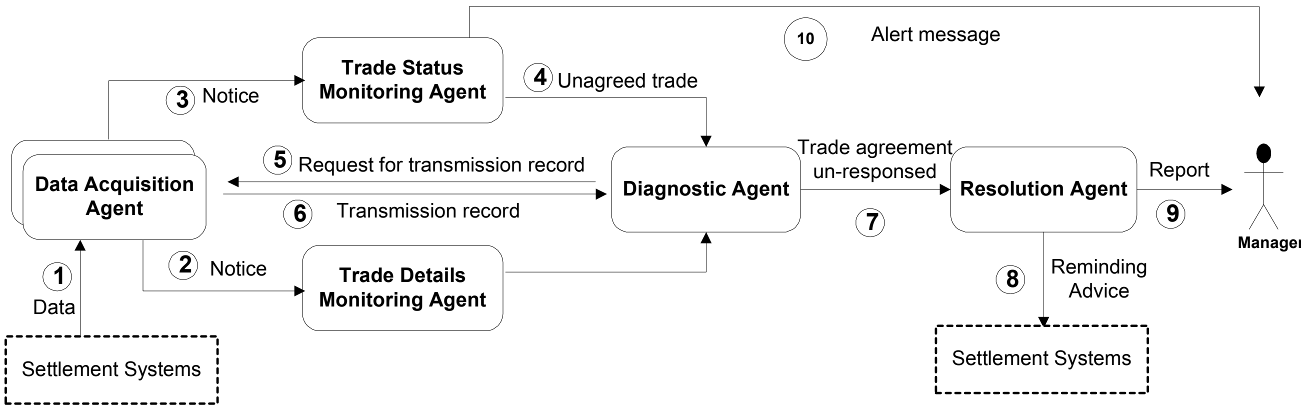

Figure 3. An Example of Agent Collaboration

In order to demonstrate how these agents work together to reach the goal of exception management, the process of their collaboration in the system is illustrated in the following example. As shown in Figure 3, the data acquisition agent keeps on capturing trade data from the settlement system based on its working schedule and data requests from other agents ①. Such autonomous activity is directed by the goal of this agent to provide data for exception monitoring and analysis. The collected data will be stored into a central repository, which works like a shared knowledge base to support the collaboration within such agents. A notice will be sent to monitoring agents to activate their monitoring activities when new data arrive ②③. When the trade status monitoring agent detects an outstanding trade agreement, i.e. a trade that has not been agreed by trade sides in a predefined time frame, it will issue an exception report to the diagnostic agent ④. In the same time, this trade exception is highlighted in the central repository for further attention. This monitoring activity is to make sure trade parties achieve agreement on trade details shortly after trade execution. Upon receiving the exception report from the monitoring agent, the diagnostic agent is activated to examine the problem. Data relevant to this trade are obtained from the central repository, and it is found that this trade is under modification due to a price error in the trade details. Based on such information and diagnostic rules in the knowledge base, the diagnostic agent may not create any resolution advice but wait for the progress of the trade for further investigation. After the trade is detected still un-agreed in a period of time after modification, the diagnostic agent will start to check the details of this trade agreement process (e.g. transmission record of the trade agreement) through the help of data acquisition agent ⑤⑥. Here, a new data acquisition agent can be activated to search additional data if the current data acquisition agent is busy in data collecting. It is found that the trade agreement has been sent out successfully without any response from the counterparty in a reasonable time. After the resolution agent receives such diagnostic report from the diagnostic agent ⑦, it may advice the settlement system to send a reminding message to the counterparty ⑧, as well as send a report to the manager ⑨. Of course, if an exception is kept unresolved for some time, the exception report with an alert message will be directly sent the to the manager for some intervention ⑩. This kind of behaviour is supported by agent's capability of continuously perceiving environment and reacting to any change from the environment based on its own knowledge and collaboration with others.

Related Work

In recent years, there have been a number of players in the securities markets to develop STP systems as well as deliver risk management and other competitive advantages as they seek to achieve T+1 settlement. Omgeo, a global joint venture of the Depository Trust & Clearing Corporation (DTCC) and Thomson Financial, has developed some solutions for straight through processing and exception processing (Omgeo 2003). Business exception notification timers identify trade components or trade sides that have been in a particular status for a defined period of time (e.g., a trade component is in an UNMATCHED status for more than 30 minutes). When this happens, Omgeo CTM generates a concise notification of the exception for retrieval by the trading partners, enabling them to take appropriate action such as amending, canceling or rejecting this trade through the client application or via the web based user interface.

SmartStream Technologies' Investigations and Reconciliations solutions enable unique integration of matching and exception management for multi-product types (SmartStream 2003). SmartStream Technologies' Investigations workflow uses flexible business rules to facilitate information flow between reconciliation and its supporting lines of business, be they internal or external customers. Its user interface organises all the data required for an investigation, including the case details, progress history, associated transactions, inbound and outbound correspondence and research documents. The Investigations monitors the progress of a case and can automatically trigger escalation procedures, such as chaser communications when responses are not received within a specified time window.

The Straight-Through Exception Processing suite (STEP™) from SunGard ePI is a common, integrated platform that promotes STP within organizations (SunGard 2003). It offers exception management automation throughout the lifecycle of a transaction, which includes the repair of pre-settlement exceptions from various systems within the STP pipe, confirmation failures, and post-settlement reconciliation and investigations. It delivers identification, assessment, and control of message-based exceptions to automatically resolve customer inquiries, payment failures, trade breaks, reconciliation failures, and STP alerts. The solution allows users to implement their organization's best practice workflows, easily incorporating what works best from a business perspective.

Conclusions and Future Work

This paper explores the approach of applying intelligent agents into exception management in securities trading. A conceptual framework of multi-agent system is designed, in which various classes of intelligent agents are proposed to provide a set of functionalities for exception management in securities trading. This kind of exception management system is independent of, and in conjunction with, existing securities trade applications. It tries to make use of internal resources to build software capabilities to interact with existing systems. This approach may reduce the cost of system development and decrease the complexity of exception resolution of STP initiatives.

By utilizing multi-agent framework, our exception management system may enjoy several benefits from software agents as follows.

1. The complex exception management tasks can be delegated to several agents on behalf of users, each carrying out a different function autonomously. Repetitive behaviors such as real time data acquisition and monitoring can benefit from agent automation and autonomy.

2. The reactive behavior of an agent allows the agent to take actions based on other agents' requests. For instance, upon receiving the exception report from a monitoring agent, the diagnostic agent will conduct data analysis and try to investigate the nature of the problem.

3. To achieve the goal of exception detection and reconciliation, agents do not simply perceive and react to changes in transaction environment, but also exhibit goal-directed behavior to take the initiative. For example, the goal of the diagnostic agent is to investigate the nature of problems. Directed by this goal, the diagnostic agent may ask help from a data acquisition agent to collect additional data if current information is insufficient to examine the problem.

4. An agent with a learning capability can learn from their environment and automatically modify their behaviors. The monitoring agents in the system may learn from the monitoring records and adjust their monitoring policy for effective detection.

5. By collaborating with each other, such agents can overcome their inherent bounds of knowledge and capability to perform complex exception detection and reconciliation tasks successfully.

In this paper, we only present an outline of the multi-agent framework. Further development of the system with detailed behaviors of each agent will be discussed in a paper in preparation. Based on this framework, we will build a prototype on the agent platform JATLite, which is developed by Stanford University. JATLite provides a set of Java templates and a ubiquitous Java agent infrastructure that makes it easy to build systems in a common way (JATLite introductory FAQ 2003). Besides the agent platform, we will adopt Jess (Java Expert System Shell) as our expert system engine. Jess is a rule engine and scripting environment written entirely in Sun's Java language (Jess 2001). Using Jess, we can build Java applets and applications that have the capacity to "reason" using knowledge which supply in the form of declarative rules. After the implementation, some simulation and evaluation will be performed on this prototype to test its performance and usability.

References

Barr, M., and Baker, J. Archer. An intelligent agent for business monitoring, in Proceedings of the IEEE Computing and Control Division Colloquium on Artificial Intelligence in Business Planning and Monitoring, 1995, IEEE Press.

Caglayan, A., and Harrison, C. Agent sourcebook: a complete guide to desktop, internet, and intranet agent, John Wiley & Sons, New York, 1997.

Financial Models Company. What is STPML? **http://www.stpml.org** 2003.

Finin, T., Richard Fritzson, Don McKay, and Robin McEntire. KQML as an agent communication language, Proceedings of the third international conference on Information and knowledge management, ACM Press, Nov.1994.

Fox, M. S., BARBUCEANU, M., and TEIGEN, R. Agent-oriented supply-chain management. International Journal of Flexible Manufacturing Systems (12), 2000, pp.165-188.

Glushko, R. J., TENENBAUM, J. M., and MELTZER, B. An XML framework for agent-based e-commerce. Communications of the ACM (42), 1999, pp.106-114.

Guerra, A. Exception Management: The Safety Net You've Been Looking For? Wall Street & Technology Online, Sep 4, 2002, URL: http://www.wallstreetandtech.com

GSCS Benchmarks. Corporate actions: Dangerous events, The 2002 Review of Major Markets, 2002, pp.20-24.

JATLite introductory FAQ. http://java.stanford.edu, **http://www-cdr.stanford.edu/ProcessLink/papers/JATL.html**, 2003.

Jess – the rule engine for the Java™ platform, **http://herzberg.ca.sandia.gov/jess**, 2001.

Jennings, N. R., and Wooldridge, M. Applications of intelligent agents: Theory and Practice, The knowledge engineering review (10:2), 1998, pp.115-152.

Kumar V., and David, G. Global Straight Through Processing. The Evolution Continues. Elec-tronic Data Systems Publication, December 2000.

Labrou, Y., FININ, T., and YUN PENG. Agent communication languages: the current landscape. IEEE Intelligent Systems (14), pp.45-52.

Securities Industry Association. Glossary, **http://www.sia.com/stp**.

Simmons, M. Securities operations: a guide to trade and position management, John Wiley & Sons, New York, 2001.

Sycara, K., PANNU, A., WILLIAMSON, M., DAJUN ZENG, and DECKER, K. Distributed intelligent agents. IEEE Expert (11), pp.36-46.

Wang, H. and Chen Wang. "Intelligent Agents in the Nuclear Industry", IEEE Computer (30:11), November 1997, pp.28-34.

Wang, H.; John Mylopoulos and Stephen Liao. Intelligent Agents and Financial Risk Monitoring Systems, Communications of the ACM (45:3), March 2002a, pp.83-88.

Wang, M. and Wang, H. Intelligent Agent Supported Flexible Workflow Monitoring System, Lecture Notes in Computer Science (2348), 2002, pp.787-791.

Wooldridge M., and Jennings N. Intelligent agents: theory and practice, The Knowledge Engineering Review (10:2), 1995, pp. 115-152.

www.epic.sungard.com/home.asp

www.omgeo.com/

www.smartstream-md.com