December 2003

Intelligent Agent Supported Exception Management in Securities Trading

Minhong Wang
City University of Hong Kong

INTELLIGENT AGENT SUPPORTED EXCEPTION MANAGEMENT IN SECURITIES TRADING[1]

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

Research Problem

With rising trading volumes in securities transactions and increasing risks faced by global financial markets, STP (Straight Through Processing) is internationally recognized by the securities industry. However, STP is not just the processing of fully automated trade and settlement cycle, but also the automation of identifying and fixing any exceptions reported in the cycle. This requires participants to enable an exception-based process to achieve early identification of errors for timely resolution (Simmons 2001). In this research we will incorporate intelligent agents into STP environment to provide automation and intelligence in exception resolution capability. A multi-agent system is to be developed, in which various classes of intelligent agents work autonomously and collaboratively to perform exception monitoring and resolution activities in securities transactions. The object of this research is to explore a novel approach to exception handling in business process management from two aspects. Firstly, with knowledge-based intelligent agents with their properties as autonomy, co-operativity, reactivity and proactivity, such systems can deliver knowledge level solution to enhance the ability of exception management in business process management. Secondly, this kind of exception management system is independent of, and in conjunction with, existing business applications, with the purpose to make use of internal resources to build software capabilities to interact with existing systems. This kind of architecture may reduce the complexity of exception resolution in current workflow systems, and applicable to business applications with or without workflow automation facilities.

Literature Review

STP and Exception Management

STP is to manage securities trades throughout the trade lifecycle automatically and without human intervention. It is only achievable if the trade information is passed within the trade lifecycle in a timely and accurate fashion. However, the major issue of exception management has not been adequately addressed. The markets get confused when they talk about exception management because they are usually talking about people being involved in the resolution (Guerra 2002). In recent years, a number of players in the securities markets have implemented STP systems and deliver risk management and competitive advantages as they seek to achieve shortened settlement cycle, such as Omgeo's Central Trade Manager, SmartStream Technologies' Investigations and Reconciliations solutions, SunGard's Straight-Through Exception Processing suite, and so on. More efforts are required to investigate the mechanism of exception resolution and improve the exception management to support STP environment.

Workflow Management and Exception Handling

Workflow management system is a promising technology aiming at the automation of business processes to improve the speed and efficiency of organizations. The unpredictability of business processes requires that workflow systems support exception handling with the ability to dynamically adapt to the changing environment, while workflow systems are currently ill-suited to dealing with exception. Nowadays exception handling is gaining increasing interests in workflow management. Researchers are exploring adaptive workflow systems in two main approaches. One track is to modify "normal" processes models to handle

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

exceptions as they occur, and the other takes partially specified process model and flexible enactment based on run time conditions (Klein et al. 2000). However most work is within the design of workflow system infrastructure, which may increase complexity to current workflow systems. Furthermore, such exception handling solutions are limited to those business applications that have employed workflow systems to support their operations.

Intelligent Agent

Intelligent agents can be seen as software agents that enjoy such properties as autonomy, co-operativity, reactivity, pro-activity and mobility [Wooldridge 1995]. The concept of intelligent agent has rapidly become an important area of research. Software agent technology provides flexible, distributed, and intelligent solutions for business process management. The benefits of applying agent technology into workflow management include distributed system architecture, easy interaction, resource management, reactivity to changes, interoperation among heterogeneous systems, and intelligent decision making (Yan et al 2001). 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 2002a, 1997).

Research Methodology

Advancement in information systems research and practice often comes from new systems concepts. However, concepts alone do not ensure a system's success. Systems must be developed in order to test and measure the underlying concepts. Systems development is therefore a key element of information systems research. Thus, we will mainly use system development methodologies (Nunamaker et al. 1991) in our research. The research process includes stating the research problems, constructing a conceptual framework, developing the system architecture, analyzing and designing the system, implementing the (prototype) system, observing and evaluating the prototype system. As to the system evaluation, some behavioural research methodologies related to questionnaire design, data gathering, data analyzing and result interpreting will be applied to test the performance and usability of the prototype system.

System Development

A multi-agent based exception management system will be developed, which is in conjunction with the legacy systems involved in securities transactions. 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 for exception management. Furthermore, the knowledge based intelligent agents are deployed to provide knowledge level solution for exception management.

Preliminary Research

We have developed a multi-agent framework for exception management in workflow management as Figure 1. It is presented in the 14th international conference on advanced information systems engineering (Wang et al. 2002). This architecture may support a flexible solution for exception handling in current workflow systems. Further research about flexible monitoring solution based on customization and knowledge learning in monitoring activities is described in a paper which has been submitted to the journal Knowledge-based Systems (Wang and Wang 2002b). In this thesis, we will apply part of this model into STP environment to explore exception resolution for securities trading.

Conceptual Framework

An exception is anything that prevents the successful completion of normal business processes. 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 2. After some investigations on possible problems in securities transactions, we found 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 2. Accordingly, we initiate our exception management solution from the monitoring activities on the three points; more details are illustrated below.

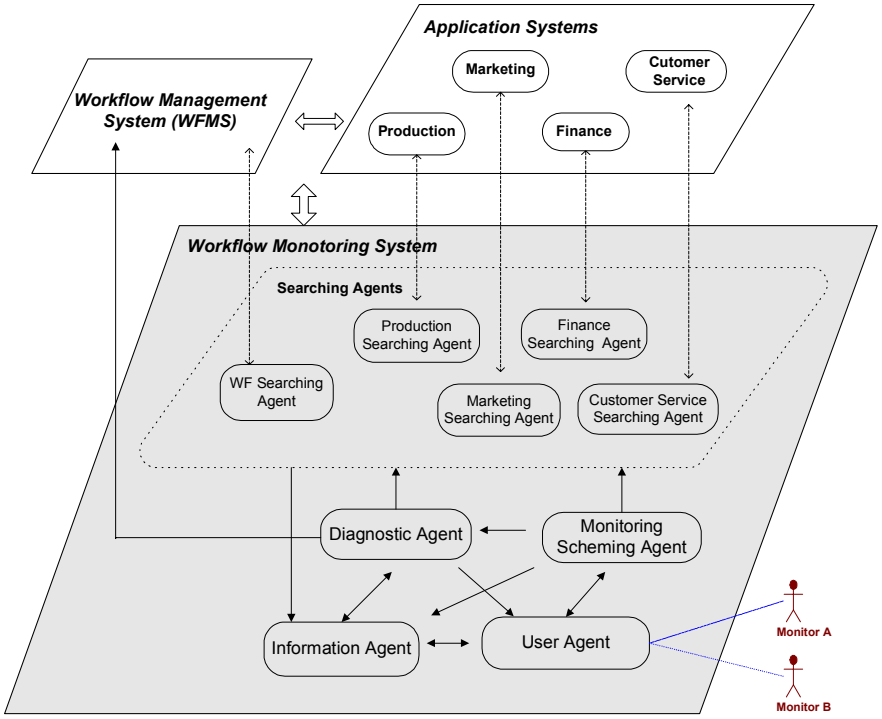

Figure 1. An Architecture of Intelligent Agents Based Workflow Monitoring System

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. Though many securities trade organizations have adopted trade validation, this activity is usually effected manually. In our system a *trade details monitoring agent* is proposed to detect such errors on a trade-by-trade base.

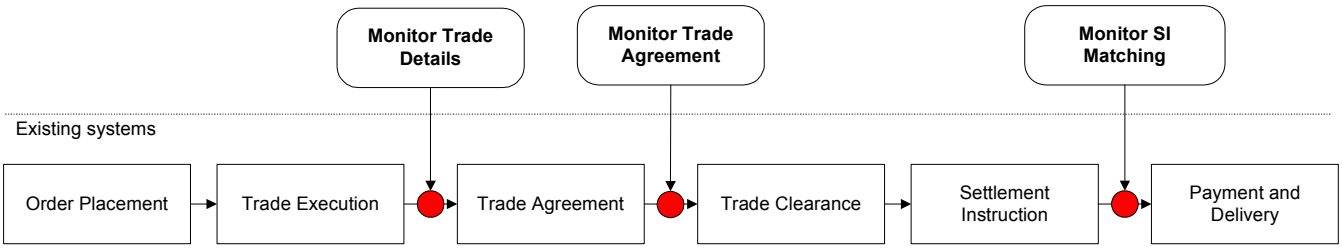

Figure 2. Trade Lifecycle with Some Monitoring Points

Monitor Trade Agreement

Trade agreement 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 being certain that the trade has been agreed with the counterparty. The issuer of the trade confirmation hopes that the recipient will check the detail upon receipt; however it is not always the case. Thus a *trade status monitoring agent* is applied to keep track of trade agreement process and highlight those un-agreed trades for further processing.

Monitor SI Matching

The matching of trade details of sellers and buyers is, in many cases, effected through two routes, namely trade agreement and SI (settlement instruction) matching. 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 who does not subscribe to a trade agreement mechanism. Similarly, we charge the *trade status monitoring agent* to detect un-matched settlement instructions.

Multi-Agent System Architecture

As outlined in Figure 3, 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 capturing real-time trade data from existing settlement system for the track on securities transactions. Two kinds of monitoring agents monitor the trading processes and keep track of exceptions. The **trade details monitoring agent** is to detect any error contained within the details of each trade, while the **trade status monitoring agent** is applied to keep detection on trade agreement and settlement instruction processes. When monitoring agents detect 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.

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.

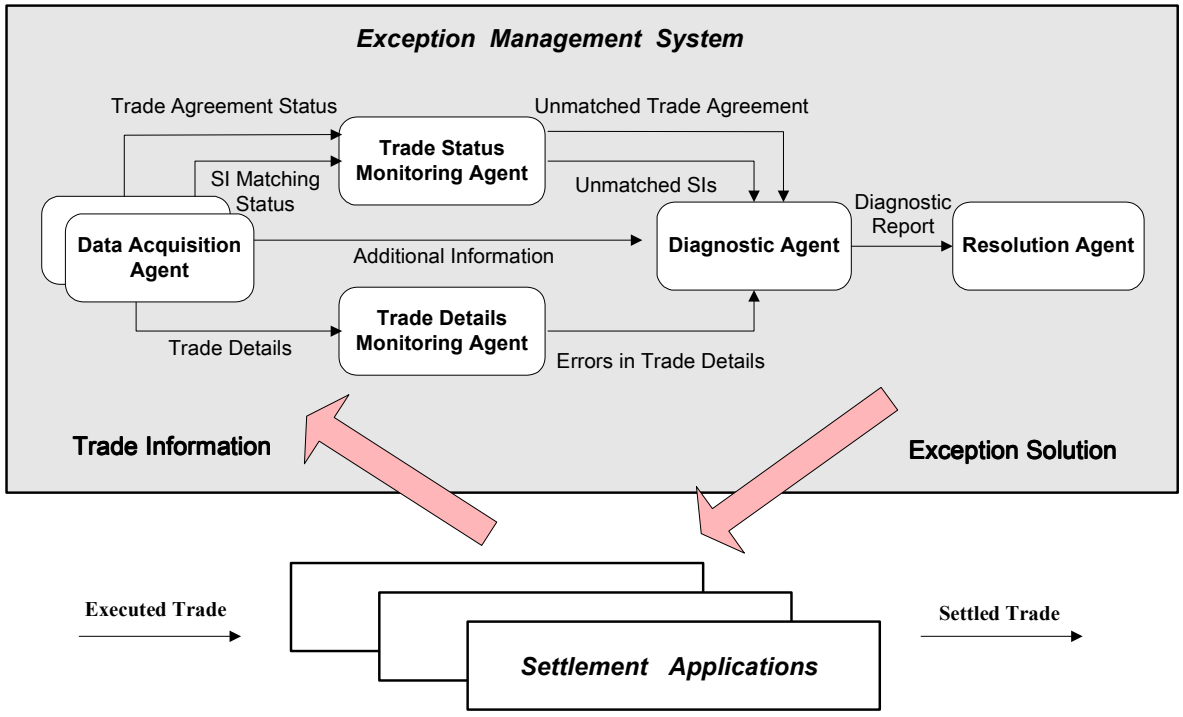

Figure 3. Multi-Agent System Architecture for Exception Management

System Design

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.

Agent Knowledge Base with Business Rule

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. It usually concerns rules, which are the user's expression of preference of policies followed by the agent to complete its task (Caglayan et al. 1997). These business rules form an important part of knowledge base of software agents. In our system, some business rules extracted from securities trading practice are applied into the design of various agents with a view to explore knowledge level solution for exception management.

Knowledge Learning

Both autonomous and semiautonomous agents rely on knowledge about the environment that has been built in by the designer. Furthermore, in today's dynamic environment of organizations and markets, the agent needs to be able to perceive its world not only for the purpose of reasoning but also for the purpose of learning. Learning is the modification of behaviour as a result of experience. In our system some agents are required to acquire their knowledge through learning. They can derive knowledge from the user and environment, and then incorporate the new knowledge into their behaviours. In rule-based agents, learning usually implies the agent's ability to automatically modify the rule base in some way. Such agents need access to historical databases or logs of events, which can be analyzed for emerging trends or correlations.

Agent Communication

The most popular language for agent communication is Knowledge Query and Manipulation Language (KQML). Recently, there are some researches focusing on the use of XML (Extensible Markup Language) in agent communication (Glushko et al. 1999). 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, Straight Through Processing Markup Language, is an XML message specification designed for the financial securities trading industry to meet the requirements of straight through processing (Financial Models Company 2003). Agents will send and receive information through XML encoded message with KQML like format. Part of STPML is adopted as the information model to represent securities transaction data.

Prototype Implementation

The prototype of this multi-agent system will be built in Java so that agents can run on heterogeneous platforms and make use of lightweight applets as temporary agents. The agent platform JATLite developed by Stanford University will be used in our system. 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 also provides a template for building agents that utilize a common high-level language and protocol (JATLite introductory FAQ 2003). Besides the agent platform, we adopt Jess (Java Expert System Shell) as our business rule engine. Jess is a rule engine and scripting environment written entirely in Sun's Java language by Ernest Friedman-Hill at Sandia National Laboratories in Livermore, CA (Jess 2001). By 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.

Prototype Evaluation

According to (Delone et al. 1992), there are six major factors that contribute to information systems success – system quality, information quality, information use, user satisfaction, individual impact, and organizational impact. These categories are interrelated and interdependent, forming IS success model. In this research, we will choose two categories to evaluate our prototype. The first is "System Quality", which is to measure the information system itself based on such items as data accuracy, data currency, system flexibility, and so on. The second is "User Satisfaction", which can be measured by overall satisfaction, satisfaction of exception detection, satisfaction of exception resolution, etc. Some simulations or experiments can be performed on the prototype to test its performance and usability. In such simulations and experiments some treatments will be employed as factors on system performance. For example, we may test whether the system with agent support works better than the same system without agent support, or test the effect of different levels of agent knowledge (e.g. with or without knowledge learning). These factors, which are under the control of the experimenter, will be used as independent variables to test their influence on dependent variables related to the system performance.

Research Progress and Future Work

This research is to incorporate intelligent agents into securities trading environment to provide automation and intelligence in exception resolution capability. It attempts to explore some possible approaches to facilitate exception handling in business process management. So far we have finished system architecture and part of system design; the main work described in this paper has been accepted by AMCIS 2003. After detailed design we will start to build a prototype and perform some evaluation on the prototype. In the future we will extend our current research to web services. Web Services are loosely coupled software components delivered over Internet standard technologies. The term "Web Services" represents a standard-based way of creating applications that can work cooperatively with other applications, regardless of the platform or program languages used between them (Clabby 2003). Web services will play a role in STP of business process management though it is still far away. We plan to develop intelligent agent supported web services for exception management in securities trading. For instance, a trade validation web service can be integrated with a securities trading organization's legacy system to help detect and correct errors in trade details. A settlement monitoring web service can de deployed for an existing settlement system to keep track on settlement progress. Furthermore, based on such web services, we can develop high-level application to support overall supervision on securities transactions.

References

Caglayan, A., and Harrison, C. *Agent Sourcebook: AComplete Guide to Desktop, Internet, and Intranet Agent*, John Wiley & Sons, New York, 1997.

Clabby, J. *Web Services Explained Situations and Application for the Real World*, Prentice Hall, Upper Saddle River, NJ,, 2003.

Delone, W.H., and Mclean, E.R. "Information Systems Success: The Quest for the Dependent Variable," *Information Systems Research* (3:1), 1992, pp. 60-85.

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

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**.

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.

Klein, M., Dellarocas, C., Bernstein, A. "Introduction to the Special Issue on Adaptive Workflow Systems," in *Proceedings of the Fifth International Conference Computer Supported Cooperative Work (CSCW)*, November 2000, pp. 265-267.

Nunamaker, J.F., Chen, M., Purdin, T.D.M. "Systems Development in Information Systems Research," *Journal of Management Information Systems* (7:3), Winter 1990-1991, pp. 89-106.

Simmons, M. *Securities Operations: A Guide to Trade and Position Management,* John Wiley & Sons, New York, 2001.

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

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

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

Wang, Minhong and Wang, Huaiqing. "Intelligent Workflow Monitoring with Agent Technology," submitted to *Knowledge-Based Systems* (2002b).

Wooldridge, M., and Jennings, N. "Intelligent Agents: Rheory and Practice," *The Knowledge Engineering Review* (10:2), 1995, pp. 115-152.

Yan, Y., Maamar, Z., and Shen, W. "Integration of Workflow and Agent Technology for Business Process Management," in *Proceedings of the Sixth Inbternational Conference on Computer Supported Cooperative Work in Design,* 2001, pp. 420-426.