The Agent Reputation and Trust (ART) Testbed Architecture

Karen K. Fullam[1], Tomas B. Klos[2], Guillaume Muller[3], Jordi Sabater[4], Zvi Topol[5], K. Suzanne Barber[1], Jeffrey S. Rosenschein[5], and Laurent Vercouter[3]

[1] Laboratory for Intelligent Processes and Systems, University of Texas at Austin, USA
[2] Center for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands
[3] École Nationale Supérieure des Mines, Saint-Étienne, France
[4] Institute of Cognitive Science and Technology (ISTC), National Research Council (CNR), Rome, Italy
[5] Multiagent Systems Research Group—Critical MAS, Hebrew University, Jerusalem, Israel

Abstract. The Agent Reputation and Trust (ART) Testbed initiative has been launched with the goal of establishing a testbed for agent reputation- and trust-related technologies. This testbed serves in two roles: (1) as a competition forum in which researchers can compare their technologies against objective metrics, and (2) as a suite of tools with flexible parameters, allowing researchers to perform customizable, easily-repeatable experiments. In the testbed's art appraisal domain, agents, who valuate paintings for clients, may gather opinions from other agents to produce accurate appraisals. This paper first gives a brief overview of the testbed domain problem to orient the reader to the game rules. A detailed discussion of the ART Testbed implementation architecture is presented, explaining the functionality of the testbed's Game Server, Simulation Engine, Database, User Interfaces, and Agent Skeleton.

1 Introduction

The Agent Reputation and Trust (ART) Testbed [8] initiative has been launched with the goal of establishing a testbed for agent reputation- and trust-related technologies. This international team of eight researchers from five countries has been formed to coordinate domain design, game specification, testbed development, and competition administration. The ART Testbed is designed to serve in two roles: (1) as a competition forum in which researchers can compare their technologies against objective metrics, and (2) as a suite of tools with flexible parameters, allowing researchers to perform customizable, easily-repeatable experiments.

In recent years, researchers [1, 4, 7] have recognized objective standards are necessary to justify successful trust modeling systems and provide a baseline of certifiable strategies for future work. For trust algorithms and representations to crossover into application [2, 3], the public must be provided with system evaluations based on transparent, recognizable standards for measuring success. As a versatile, universal experimentation site, the ART Testbed will foster a cohesive exploration of trust research problems; researchers will be united toward a common challenge, out of which can

come solutions to these problems via unified experimentation methods. Through objective, well-defined metrics, the testbed will provide researchers with tools for comparing and validating their approaches. The testbed will also serve as an objective means of presenting technology features—both advantages and disadvantages—to the research community. In addition, the ART Testbed will place trust research in the public spotlight, improving confidence in the technology and highlighting relevant applications.

In the testbed's art appraisal domain, agents, who valuate paintings for clients, may gather opinions from other agents to produce accurate appraisals. This paper first gives a brief overview of the testbed domain problem in Section 2 to orient the reader to the game rules. In Section 3, a detailed discussion of the ART Testbed implementation architecture is presented, explaining the functionality of the testbed's Game Server, Simulation Engine, Database, User Interfaces, and Agent Skeleton. Finally, Section 4 concludes, presenting future plans, including upcoming testbed code releases and competitions.

2 Testbed Domain Problem

The testbed operates in two modes: competition and experimentation. In competition mode, the testbed compares different researchers' strategies as they act in combination. Each participating researcher controls a single agent, which works in competition against every other agent in the system. Competition organizers can change parameters to permit the game structure to be adapted for subsequent competitions. To utilize the testbed's experimentation mode, the complete testbed will be downloadable for researcher use independent of the competition. Thus, results may be compared among researchers for benchmarking purposes, since the testbed provides a well-established environment for easily-repeatable experimentation. In experimentation mode, the researcher has the flexibility of complete control over all experiment parameters.

In the art appraisal domain, agents function as painting appraisers with varying levels of expertise in different artistic eras (e.g. classical, impressionist, postmodern). Clients request appraisals for paintings from different eras; if an appraiser does not have the expertise to complete the appraisal, it can purchase opinions from other appraisers. Other appraisers estimate the accuracy of opinions they send by the cost they choose to invest in generating an opinion; opinion providers may lie about the estimated accuracy of their opinions. Appraisers produce appraisals using opinions they receive from other appraisers; they receive more clients, and thus more profit, for producing more accurate appraisals. Appraisers may also purchase reputation information from one another, which helps appraisers know from which other appraisers to request opinions. More detail about the art appraisal domain rules can be found in the ART Testbed Competition Rules [5]. In [6], the authors detail today's most prominent trust research objectives, solutions to which the testbed must facilitate, and present a structured defense of the chosen art appraisal domain by delineating how the domain problem addresses each of the prominent research objectives.

Appraiser strategies are analyzed in terms of both agent- and system-based metrics. The agent perspective examines the utility of a strategy to a single appraiser without regard for the benefit to the overall agent system. In the testbed's appraisal scenario,

appraisers attempt to accurately valuate their assigned paintings; their decisions about which opinion providers to trust directly impact the accuracy of their final appraisals. Therefore, in competition mode, the winning agent is selected as the appraiser with the highest bank account balance. In other words, the appraiser who is able to (1) estimate the value of its paintings most accurately and (2) purchase information most prudently, is deemed most successful. The testbed also provides functionality to compute the average accuracy of the appraiser's final appraisals and the consistency of that accuracy, represented as its final appraisal error mean and standard deviation, respectively. In addition, the quantities of each type of message passed between appraisers are recorded.

The system perspective employs metrics that emphasize social welfare, or benefit to the appraiser network as a whole. The testbed collects data on system metrics such as distribution of earnings among appraisers, number of messages passed (by type), number of transactions, and transaction distribution across appraisers. Finally, the testbed provides methods allowing researchers to define additional metrics and collect relevant data.

3 Implementation Architecture

As shown in Figure 1, the testbed architecture, implemented in Java, consists of five components:

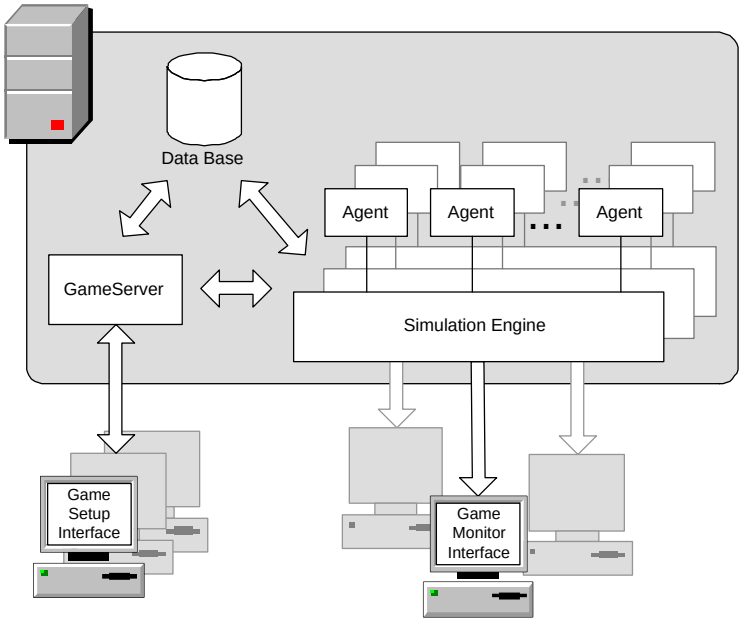


Fig. 1. ART Testbed architecture.

1. Game Server, which allows setting up and running games (Section 3.1),
2. Simulation Engine (one per game), which controls the simulation environment (Section 3.2),
3. Agent Skeleton, designed to assist players in implementing their strategy code (Section 3.3),
4. Database, which stores data for Simulation Engine calculations and experiment analysis (Section 3.4), and
5. User Interfaces (Game Setup Interface and Game Monitor Interface), through which games are set up and viewed (Section 3.5).

In competition mode, the Game Server, Simulation Engines, and Database run on a central machine maintained by the ART Testbed Team, while the User Interfaces are locally accessible to players and observers. In experimentation mode, the researcher may run the Game Server, Simulation Engines, and Database locally. The following subsections provide detailed descriptions of each module.

3.1 Game Server

The Game Server manages the scheduling and starting of all games. A game is initiated by specifying parameters for new game through the Game Setup Interface (Section 3.5), either by the ART Testbed Team (in competition mode) or by the researcher (in experimentation mode); the Game Server records these game parameters in the Database and initiates a Simulation Engine (Section 3.2) for each new game. A properties file, input as the Game Server is instantiated, determines which game parameters are considered 'fixed' and which may be defined by the user when initiating a game. The Game Server also registers Agents (Section 3.3) for games, recording agent data in the Database as well.

3.2 Simulation Engine

A Simulation Engine is responsible for controlling a single game's simulation environment by enforcing chosen parameters. The Simulation Engine also assigns clients to appraisers and coordinates communication among appraisers. In each period, the Simulation Engine manages allocation of clients to appraisers, the conducting of opinion and reputation transactions, and calculation of final appraisals, as demonstrated in Figure 2.

Specifically, a single game proceeds as a series of periods. In each period, the Simulation Engine coordinates the following actions in sequence:

1. At the start of each period, the Simulation Engine assigns clients (with paintings to be appraised) to each appraiser. Appraisers receive larger shares of clients if they have produced more accurate appraisals in the past, to simulate clients' preference for accurate appraisers. The Simulation Engine sends appropriate client appraisal request notifications to assigned appraisers to simulate clients requesting appraisals. To simulate the payment of up-front appraisal fees by clients, the Simulation Engine deposits fees into appppropriate appraiser bank accounts.

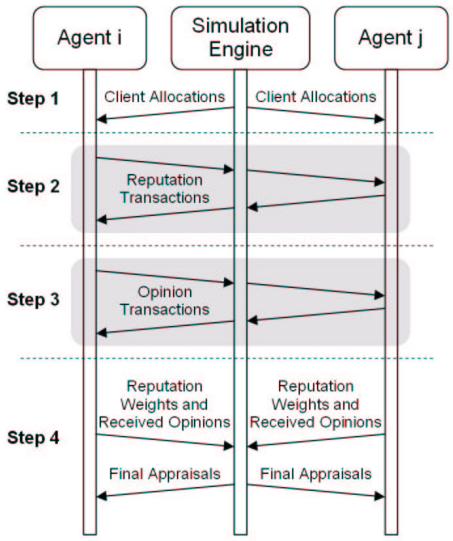
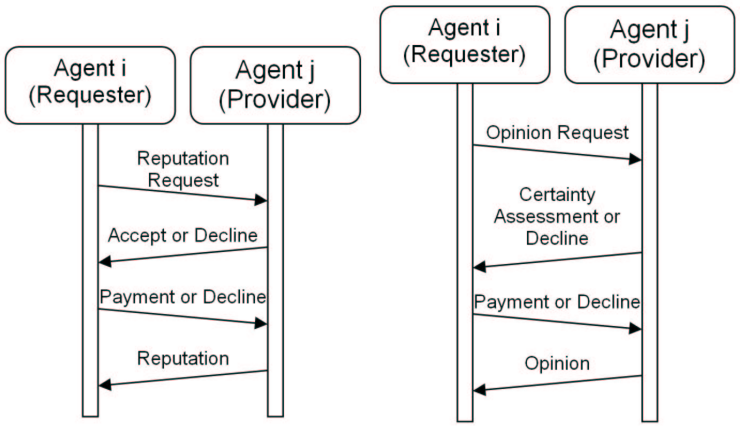


Fig. 2. Simulation Engine tasks for a single period, including: allocating clients (Step 1), conducting reputation transactions (Step 2), conducting opinion transactions (Step 3), and calculating final appraisals (Step 4).

2. Next, the Simulation Engine facilitates reputation transactions among appraisers. Appraisers may conduct reputation transactions about third-party appraisers' expertise in given eras to decide from which appraisers to purchase appraisal opinions. Figure 3(a) shows the sequence of steps in a reputation transaction. If a potential reputation provider chooses to accept an appraiser's reputation request, the requesting appraiser sends payment and the provider sends the reputation, which need not be truthful. Alternatively, the provider may reject the request, and the transaction is aborted. The Simulation Engine serves as a go-between for agent communication, handling the passing of transaction messages. In addition, the Simulation Engine handles payments between appraisers by withdrawing from requester bank accounts and depositing into provider bank accounts.
3. The Simulation Engine then faciliates opinion transactions among appraisers. Figure 3(b) shows the sequence of steps in an opinion transaction. When an appraiser requests an opinion, the potential provider may reject the request (thus the transaction is aborted), or send a (possible untruthful) certainty assessment about the opinion it claims it will send. In the latter case, the requester may either decline the potential opinion (and abort the transaction) or send payment. Upon receipt of payment, the provider sends the opinion, which need not be truthful. Again, the Simulation Engine handles the passing of transaction messages and payment withdrawals and deposits.
4. Finally, the Simulation Engine calculates the appraiser's final appraisal as a weighted average of the opinions the appraiser purchases. Opinion weights are real values be-



(a) Reputation transaction protocol.

(b) Opinion transaction protocol.

Fig. 3. Reputation and Opinion Transaction Protocols.

tween zero and one that an appraiser assigns, based on its trust model, to another's opinion. The Simulation Engine, not the appraiser itself, performs the appraisal calculation to ensure that the appraiser does not employ strategies for selecting opinions to use after receiving all purchased opinions.

While conducting each of the above tasks, the Simulation Engine passes all relevant data to the Database for recordkeeping, as described below in Section 3.4. The data is then available for retrieval to complete necessary calculations.

3.3 Agent Skeleton

The Agent Skeleton is designed to allow researchers to implant within their appraiser agent-customized trust representations and algorithms while permitting standardized communication protocols with other entities. All appraiser agents participating in the ART Testbed are descendants of the same abstract class `Agent`. This class defines a set of abstract methods to be coded by the researcher to define the behavior of his/her appraiser agent, as well as a set of methods to facilitate the communication with other appraiser agents. The `Agent` class also provides methods for interacting with the Simulation Engine (for tasks such as verifying bank balances).

Researcher-Coded Methods The following abstract methods must be implemented to give the agent competitive functionality in the ART Testbed.

- `void prepareReputationQueries()` The agent must determine which other agents to query for reputations and the agents about whom to request. The agent can send as many queries as desired.
- `void prepareReputationResponses()` Depending on received reputation queries, the agent must decide whether to respond and, if responding, whether to respond truthfully.
- `void prepareOpinionRequest()` The agent must decide about which paintings to seek opinions from other appraiser agents and which appraisers to query.
- `void prepareCertaintyAssessment()` Depending on received opinion requests, the agent must decide whether to respond and, if responding, what certainty assessment to provide to the requester (certainty assessments may be falsified).
- `void confirmOpinionRequest()` After analyzing received certainty assessments, the agent must decide whether to confirm purchase of opinions proposed by potential providers.
- `void processOpinionOrder()` To satisfy an opinion request, the agent must decide what opinion value to return to the requester.
- `public ArrayList reportWeights()` This method is called by the Simulation Engine when calculating an agent's final appraisal. The agent must report a reputation weight for each opinion it expects to receive.
- `void processFeedback()` Upon learning a true painting value, the agent must revise its trust models of opinion-providing appraisers.

Methods for Inter-Agent Communication The communication between appraiser agents is based on a message-passing scheme handled by the Simulation Engine. Messages are implemented as Java classes specifying the message sender, message receiver, and a conversation thread identifier (the transaction conversation to which the message belongs). Table 3.3 shows the different types of messages used in an ART scenario.

ReputationQuery	The agent requests the reputation of a third-party appraiser in a given era.
ReputationResponse	The agent provides a reputation in answer to a reputation query.
OpinionRequest	The agent requests an evaluation opinion of a painting.
CertaintyAssessment	The agent responds to an opinion request by communicating its skill level for the required era.
OpinionOrder	The agent confirms the request for an opinion about which certainty has been communicated.
Opinion	The agent communicates an opinion about a painting.

Table 1. Types of messages

The abstract class `Agent` provides the method `void sendMessage (Message m)` that sends a message to the corresponding agent (as specified inside the message 'm'). The agent first has to instantiate a message class of the desired type, to fill its

properties, and to call the `sendMessage` method. The message-sending task is further facilitated by a set of methods for creating and sending specific message types in a single step.

Message receipt is performed by calling the method `Iterator getMessages()`, which returns an Iterator of the relevant received-message list. For example, if the `getMessage` method is called inside the method `prepareReputationResponses()`, the agent will get an Iterator over a list of the reputation queries that need to be considered.

Each agent maintains two internal message lists managed by the Simulation Engine: an InBox (incoming messages) and an OutBox (outgoing messages). Calling the `sendMessage` method adds a message to the OutBox list. The Simulation Engine collects all messages in an agent's OutBox, checks the validity of each message, and adds each message to the receiver's InBox. Invalid messages (for example, with no receiving agent designated), are deleted. Also, only the messages with the correct type given the current state of the simulation are accepted.

3.4 Database

The Database stores all information about the games. First, it stores information about the participants (identification and classes provided) and the parameters for the current games. Then, during the course of these games, the Database collects, through the Simulation Engine, environment and appraiser data. These data are stored in a MySQL database and include:

1. For each appraiser for each timestep:
   - client share,
   - fees paid to or from the agent (for reputations, opinions, or client appraisals),
   - bank account balance,
   - reputation weights associated with other appraisers,
   - generated opinions,
   - calculated final appraisals.
2. For the environment for each timestep:
   - true painting values,
   - all messages exchanged between appraisers.

In addition, the testbed is designed to provide researchers with the functionality to log additional data types in the Database during experimentation mode. With access tools for navigating Database logs, data sets are made available to researchers after each game session for game re-creation and experimental analysis. Information stored in the Database is extracted by the Simulation Engine to populate the User Interfaces, as detailed below in Section 3.5.

3.5 User Interfaces

User Interfaces permit the creation of games (according to user-specified parameters) and the joining of agents to games. The User Interfaces also allow observation of games

in progress and access to information collected in the Database by graphically displaying details such as transactions between appraisers, accuracy of appraisers' final appraisals, and money earned by each appraiser. In competition mode, the ART Testbed Team (competition organizers) initiate official competition games with pre-specified parameters, while competition participants may join and view those official games. In experimentation mode, researchers may initiate games (with the parameters of their choosing), join games, and view games according to experiment needs.

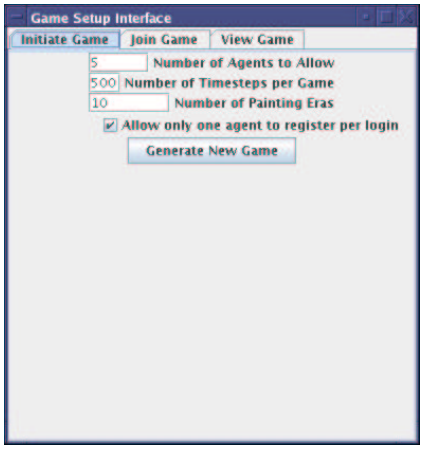
To interact with the testbed, the user first launches the Game Setup Interface (Figure 4). To initiate a new game (Figure 4(a)), the user specifies game parameters, such as the number of agents to allow. Upon clicking `Generate New Game`, a new game is initiated, and a game name and start time are assigned.

To join a game that has already been initiated (Figure 4(b)), perhaps by another user, the user selects from the initiated games (with start times listed), then names each agent and selects the agent's class file to submit to the simulation. Games may only be joined between the time at which the game is initiated and the time at which the game is scheduled to start (agents may not join a game after the game as begun). The user enters a login and password to secure viewing access to the agents owned by the user.

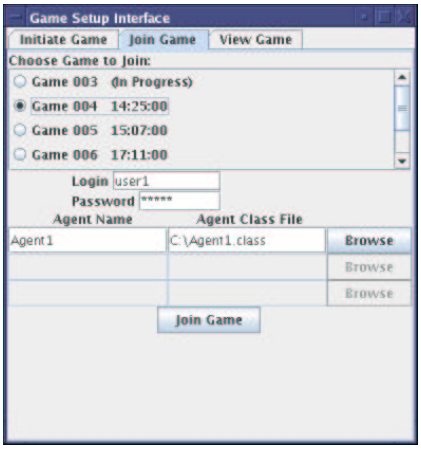
A user can view any initiated game (Figure 4(c)) as a guest by simply selecting the game to view. By entering a login and password, a user is granted permission to view additional game results related to the agents owned by the user. Clicking `View Game` launches the Game Monitor Interface. Clicking `Download Game Data` permits the user to save the data of a completed game, including private data of agents owned by the user, as an XML file.

The Game Monitor Interface, shown in Figure 5, consists of a topology panel on the left, displaying the entire agent system, and a set of detail panels on the right. The user can view a detail panel of in-depth statistics for each agent owned by the user. In addition, a `System Data` detail panel displays additional system-wide statistics. The left-hand topology panel shows the current client share (proportionally represented by the number of client figures in the corresponding blue oval) and bank balance of each agent, updated after each simulation period. Arrows show the direction and quantity of each type of message passed between agents, including opinion requests, opinions, reputation requests, and reputations. Arrow thickness is proportional to message volume.

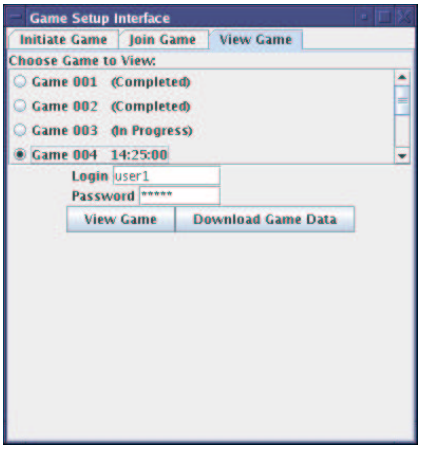
An agent's detail panel (Figure 6) shows numerous statistics to assist the user in gauging the agent's performance. The `Bank Balance` chart (Figure 6(a)) measures the agent's total bank balance after each period. The `Transaction Fee` chart displays fees the agent earns or pays in each period, such as appraisal fees earned from clients and opinion costs paid to opinion providers. An overlaid line displays the total amount earned or lost in each period. The user can choose which subset of fees to display on the chart. Furthermore, the user can view the reputation weights the agent has estimated for each other agent in the system with the `Reputation Weights` chart. Scrolling down the agent's detail panel (Figure 6(b)) The `Average Appraisal Error` chart displays the agent's accuracy, with standard deviation error bars, in estimating appraisals for clients. In the `Transactions` chart, the user can view per-period counts of transactions (opinions and reputations sold to or purchased from an-



(a) Initiate Game panel.



(b) Join Game panel.



(c) View Game panel.

Fig. 4. Game Setup Interface panels for initiating, joining, and viewing games.

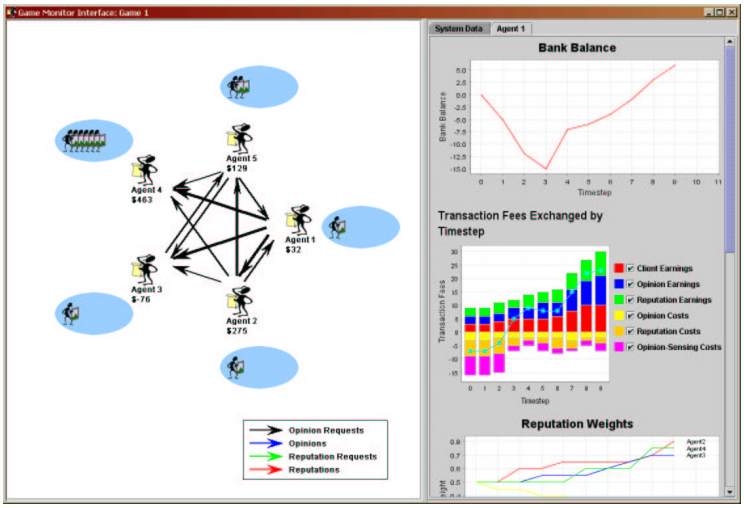


Fig. 5. Game Monitor Interface.

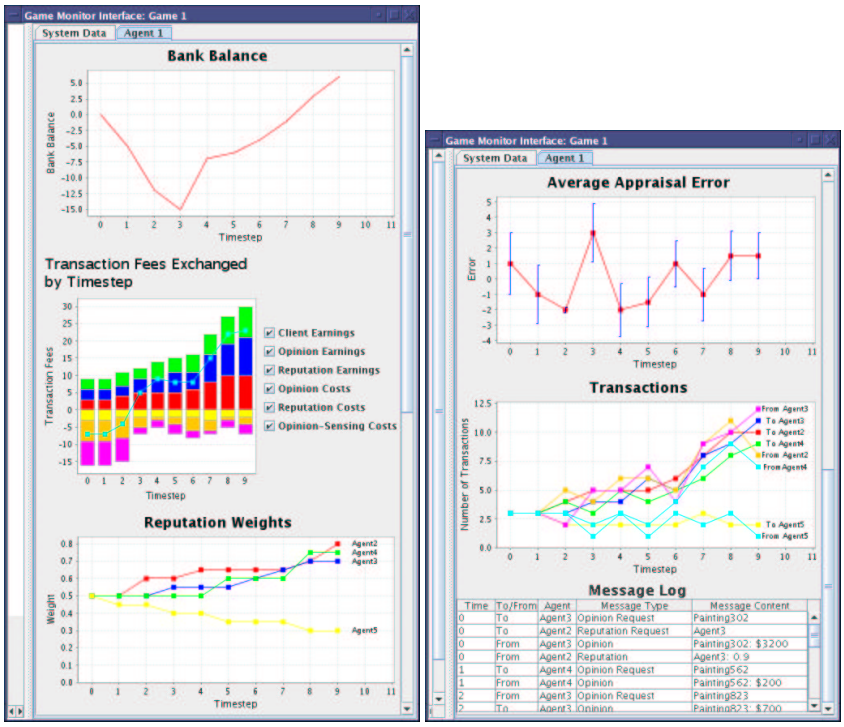
other agent). The `Message Log` table at the bottom of the detail panel displays a log of every message sent or received by the agent.

The `System Data` detail panel (Figure 7) provides statistics about all agents in the system. The user can compare bank balances and average appraisal errors among all agents (Figure 7(a)). Scrolling further down the `System Data` detail panel (Figure 7(b)), the user can view statistics comparing the number of completed transactions (both opinion and reputation transactions) between each pair of agents in the system. The `Message Log` table on the `System Data` detail panel logs every message passed in the system.

4 Conclusions

This paper details the implementation architecture for the ART Testbed, which provides researchers with easy access to a common experimentation environment and allows researchers to compete against one another to determine the most viable technology solutions. The testbed provides functionality through five modules—Game Server, Simulation Engine, Database, User Interfaces, and Agent Skeleton—to promote easy experimentation and agent development by participating researchers.

The testbed team plans to demonstrate the testbed in July of 2005, releasing a beta version of the testbed shortly after. Based on prototype testbed experimental review and feedback from the research community, the first testbed competition will be conducted in July of 2006. Development progress can be monitored through the ART Testbed website [8], where updates to testbed development are posted periodically. The website also hosts a discussion group through which the trust research community can provide feedback.



(a) Bank balance, transaction fee charts, and reputation weights.

(b) Transaction chart, message log table and average appraisal error charts.

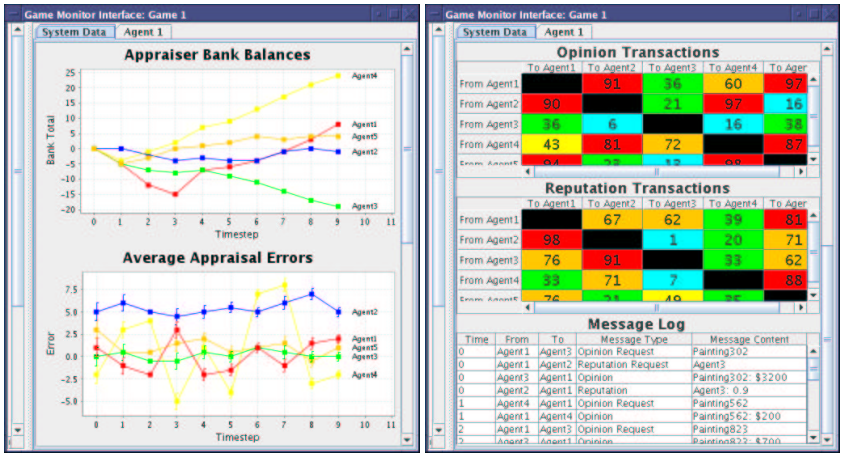
Fig. 6. The `Agent Data` panel.

Acknowledgment

Jordi Sabater enjoys a Sixth Framework Programme Marie Curie Intra-European fellowship, contract No. MEIF-CT-2003-500573. Tomas Klos is working on the CIM-project on Cybernetic Incident Management, sponsored by the Dutch government (SENTER) under project number TSIT2021. Research by Karen Fullam is sponsored in part by the Defense Advanced Research Project Agency (DARPA) Taskable Agent Software Kit (TASK) program, F30602-00-2-0588.

References

1. K. Barber, K. Fullam, and J. Kim. Challenges for Trust, Fraud, and Deception Research in Multi-agent Systems. In *Trust, Reputation, and Security: Theories and Practice*, volume 2631 of *LNCS*, pages 8–14. Springer, 2003.



(a) Bank balance and average appraisal error charts.

(b) Transaction chart and message log table.

Fig. 7. The `System Data` panel.

2. BizRate. *BizRate*. http://www.bizrate.com, 2002.
3. eBay. *eBay*. http://www.eBay.com, 2002.
4. K. Fullam and K. S. Barber. Evaluating Approaches for Trust and Reputation Research: Exploring a Competition Testbed. In M. Paolucci, J. Sabater, R. Conte, and C. Sierra, editors, *Proc. IAT Workshop on Reputation in Agent Societies*, pages 20–23, 2004.
5. K. Fullam, T. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. S. Rosenschein, L. Vercouter, and M. Voss. The Agent Reputation and Trust (ART) Testbed Competition Game Rules (version 1.0). Technical report, The University of Texas at Austin TR2004-UT-LIPS-028, 2004.
6. K. Fullam, T. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. S. Rosenschein, L. Vercouter, and M. Voss. A Specification of the Agent Reputation and Trust (ART) Testbed. In *Proc. AAMAS*, 2005.
7. J. Sabater. Toward a Test-Bed for Trust and Reputation Models. In R. Falcone, K. Barber, J. Sabater, and M. Singh, editors, *Proc. of the AAMAS-2004 Workshop on Trust in Agent Societies*, pages 101–105, 2004.
8. ART Testbed Team. *Agent Reputation and Trust Testbed*. http://www.lips.utexas.edu/˜kfullam/competition/, 2005.