A Testbed for the Evaluation of Multi-Context Trust Models in Open Multi-Agent Systems

Mifrah Youssef [1], En-Nouaary Abdeslam[1] and Dahchour Mohamed [1]
[1] Institut National des Postes et Télécommunications, Rabat, Morocco.

Summary
In open dynamic multi-agent systems, trust is commonly considered as a critical concept to be handled and managed. Computational trust models are a kinds of formal models that have been proposed to manage trust in such situation. These models present a new form of distributed intelligence in virtual societies and collective intelligence. However, the diversity of those models makes user confused about which one to choose. Different testbeds were proposed to evaluate trust and reputation systems and verify the robustness of the underlying trust models. However, those testbeds are not flexible to handle different scenarios in various contexts. In this paper, the authors present a testbed for evaluating computational trust models that could provide user more flexibility while comparing trust models in open systems. The ultimate objective is to evaluate and classify available computational trust models.

Key words:
trust, reputation, testbed, multi-agent systems.

1. Introduction

Trust is an ubiquitous concept that exists in our social world. The concept has become a booming topic of research due to the fast growing that knows the area of distributed architectures and multi-agent systems. Multi-agent systems are a kind of distributed systems that consist of multiple interacting intelligent agents, used together to solve problems which are difficult to be solved by an individual agent. An agent that belongs to a multi-agent system is able to communicate and act within the system and analyze perceived events using its own strategy. In the last two decades, multi-agent systems and distributed architectures have been converted to a more open structure with less restriction on the internal behavior of agents in the system. Many systems in use by millions of users today provide such features. P2P networks, online games, social networks and e-commerce platforms are a kind of open dynamic networks, where users could enter and leave the system dynamically. With the evolution that knows the area of those open distributed systems, a new kind of malicious intelligent agents could be developed and used in those systems. Malicious agents have the intention to switch their behaviors and to act dishonestly. This kind of intelligent agent raises a challenge of managing trust and reputation within the system. To address the trust and reputation challenge in such situation, researchers have looked for a formalism of the trust management so that an agent can apply formal strategies to decrease the risk of delegating tasks to distrusted peers. Those researches led to computational trust models, where their authors propose metrics, and learning strategies for trust assessment that could be applied by agents to manage and evaluate the trustworthiness of their peers. Learning strategies proposed by a trust model wrap the intelligence that an agent will use while managing trust within the system. When using a computational trust model within a distributed system, agents that belong to this system will use one or more learning strategies proposed by the model. Each of the trust models, found in the literature, presents its main components and its specific learning strategies to use for managing trust. When introducing a trust model into an implementation's system, agents that belong to this system become able to reason about trustworthy of their peers. The diversity of computational trust models makes the user confused while trying to select a model for a specific system. Another difficulty that arises is that the proposed models are experimented using limited scenarios and a set of data proposed by their authors. To address this situation, comparative tools called testbeds were proposed by researchers to expose those computational trust models to different scenarios, and also to compare the robustness of each model with respect to different scenarios. Those testbeds have shown their utilities of verifying the perception level of trust models. But the lack in existing testbeds is that some of them do not handle trust per context, others do not handle diverse attack strategies. To this end, the authors propose a testbed that presents two advantages: on the one hand, the proposed testbed manages trust assessment per context while handling diverse services, and on the other hand it introduces various attack strategies to evaluate the robustness of trust models against those attacks.

The remainder of this paper is organized as follows. Section 2 presents some of the known existing testbeds proposed to evaluate computational trust models. Section 3 is an overview of a proposed Framework for Modeling Multi-Agent Trust denoted for (GeFMMAT) and it's Meta-Model. Section 4 presents the implementation of GeFMMAT using JADE. Section 5 introduces the proposed testbed. It describes the architecture and also the implementation of the testbed along with some

Manuscript received July 5, 2017
Manuscript revised July 20, 2017

experimentation. Finally, Section 6 concludes the paper and presents perspectives for future work.

2. Trust Testbeds : Related Works

To handle the diversity of existing trust models, some testbeds have been proposed to evaluate and compare trust models. Four known testbeds could be found in the literature namely the Agent Reputation and Trust (ART), Trust and Reputation Experimentation and Evaluation Testbed (TREET) [7] Alpha testbed [9] and Zhang Testbed [10]. In ART, agents play the role of art appraisers. Each agent appraises the value of paintings of a client by either using its own knowledge or by asking other agents for help. Agents can use trust models to evaluate their peers and to evaluate their opinions. In ART, there is no variety of attack strategies that dishonest agents could use; there is also no collision attacks that a group of agents could perform. The ART testbed is inspired from the work of Kerr and Cohen [7] to create a more general and focused testbed, called TREET. In TREET, the game scenario is a simulation of a marketplace where sellers and buyers exchange items. Buyers are motivated by the value of items, while sellers are motivated by the profit they make from sales. Sellers can cheat by not shipping the item and so increasing their profits. With TREET, no specific format of trust value is imposed, and the user can implement new collision attacks easily. Both testbeds do not evaluate the quality of trust model classification, but just the quality of whether the selected partner is honest or not.

Another testbed, called Alpha, was proposed by Jelen [9], tries to distinct between the trust assessment phase where the agent apply learning techniques to evaluate the trustworthiness of their peers, and the decision making phase. Jelen confirm in his research that the decision making mechanism influences the performance of the trust model. Yet other recent testbed has included many attacks strategies to verify the robustness of evaluated trust models, this recent testbed was proposed by Zhang [10]. However, it still uses the scenario of buyers and sellers with a single service. This presents a lack of flexibility and could cause loss of information in the case where the evaluated trust model uses a trust evaluation per context. This raises the necessity of building a testbed that provides at the same time the possibility of managing more than one service in parallel to handle the management of trust in specific context, and to evaluate trust models against many attack strategies.

3. The GEFMMAT Framework

Several frameworks were proposed by researchers to design multi-agent systems by capturing common concepts used in such systems. Each of the proposed frameworks uses a specific domain model that design agent systems from different perspectives. Some of them are based on organizational and hierarchical perspectives, and include the notions of environment, hierarchy and role. Others focus on the interactional aspects of agent systems. There are also other works that combine between different perspectives. However, the proposed frameworks did not handle explicitly the concept of trust. To introduce the concept of trust in multi-agent systems, we have proposed a Generic Framework for Modeling Multi-Agent Systems in Untrusted Environment. This Framework is based on a meta-model that captures the semantics of concepts involved in open dynamic multi-agent systems.

Each meta-class of GeFMMAT meta-model presented in Fig.1 represents a common concept used in open dynamic multi-agent systems.

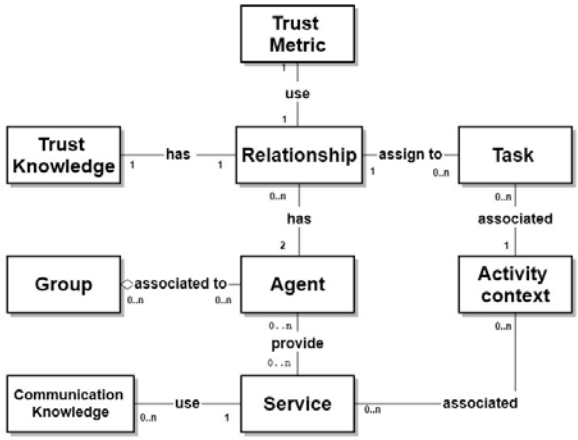

Fig.1 GeFMMAT Meta-model

The meta-class Agent presents an autonomous entity that communicates with others agents and provides one or more services. There is no constraint on the internally specification of the agent model. Features that characterize an agent within the system are defined while designing a system. An agent has a list of features values. For example, in an ecommerce platform where some users could be represented by agents, each user has a profile composed of features (user name, country, registration time, experiences, etc.). Those features constitute the agent's profile, and they reflect its instance within the system. The Service meta-class presents an expertise that an agent can propose to other agents to perform their tasks. An agent can provide one or more services. The service concept uses some communication knowledge, and it is associated to one or

more activity contexts. The Communication Knowledge presents the policy of communication. This policy could be presented by an interaction protocol and knowledge used during the communication process. It's like how the deal between the agent and his partner will be confirmed, what requirements should be shared before performing the task, and how results will be received. Those concepts would be handled by this meta-class. The Activity Context meta-class represents the boundary that group a set of common services. The Task meta-class presents the delegated task that agents exchange and delegate. It could present a request for service, or for information. A task is related to an activity context, which present the context of application of this task. The Relationship meta-class presents the information that characterizes the relationship between two agents. It's a directional relation between an initiator agent and its partners. This relationship contains also passed experiences between those agents, the trust evaluation of the initiator about its partners, and the trust metrics used. In general, if an initiator agent selects a partner to process a task, then two new relationships will be created between them, one for the initiator which presents his relation to the partner, and another one for the partner that presents his relation to the initiator. When the partner accomplishes task processing, the initiator will receive a result and use it with other parameters as an input to the trust metric to evaluate the trustworthiness of the partner. This evaluation is done using computational trust learning techniques proposed by the used model. Then, he updates the associated trust knowledge. The Trust Knowledge meta-class is defined as the model that presents the trust assessments that an agent uses and assesses while interacting with his partners. Several works have been done to setup a formal presentation for trust. Different presentation of trust have been proposed such as a variable that takes one value from a finite enumeration space , numeric variable that takes its value from a range , vector in a multidimensional space where we can set to each agent's feature an evaluation value 25 and generic form of trust using subjective logic 18. The approach of modeling trust using the subjective logic gives more control on the state of the unavailability of trust information (the case of a newcomer agent) because of the integration of the concept of uncertainty. The Trust Metric meta-class presents the logic applied to evaluate an agent. Each agent could associate a trust metric to one relationship with a partner who has accepted a delegated task. For an autonomous agent interacting with his peers, the agent could have a choice between many trust metrics. A group forms an aggregation of agents. Agents that are parts of a group could share specific features, thus, belonging to a group is helpful while identifying the nature of an agent. A group could be specified by the designer statically while designing the system, or by agents at runtime level. Those groups are dynamic, which means that members could change during the activities of the system, and at any time, the group should have an agent designated as the leader of the group. This form of aggregation would help agents to evaluate the trustworthiness of group's members more efficiently, especially when trust knowledge about the targeted agent is scarce or unavailable. The presented meta-model reflects the structural part of the framework; the behavioral part is represented by a workflow process that defines steps applied by agents to assess trust and improve the process of decision making. The framework adopts a workflow of information for the management of the trust as shown in Fig.2. This workflow captures and presents in an abstract way how agents should manage information about the environment during the decision process, and how to use feedbacks after the decision process. It defines where an intelligent agent will use the learning strategy proposed by the computational trust model, and when its trust knowledge will be updated.

Fig.2 GeFMMAT Workflow

4. JADE Based Framework Implementation

Many development platforms dedicated to implement multi-agent systems exists in the literature. There are platforms oriented middleware for implementing interoperable agent systems, such as Agent Development Kit (ADK) 12 and Java Agent DEvelopment Framework (JADE). There are social platforms that handle the organizational architecture and help expressing group behaviors such as MadKit. There are also reasoning platforms that focus on the internal processing of agents within the systems like JASON [22] and SOAR [24]. Pokahr et al. established in a detailed research [23] a classification of existing platforms. Each platform is based on a set of standards and specifications such as FIPA [11], JXTA [25], and web services. The advantage of the JADE over other platforms is that it complies with the FIPA specification for interoperable intelligent multi-agent systems and represents an agent middleware. It uses also an agent abstraction to design agent in the system. The

JADE platform provides a set of graphical tools to be used during the development process. The established implementation of the proposed Framework is based on JADE. We have extended the jade.core.Agent class to an Agent class that presents an abstraction of the agent used in our Framework. We have also introduced new classes such as AgentExperience that presents the acquired experience and TaskHandler that presents a listener for task request. A trust model metric is presented by the interface ITrustMetric where trust learning algorithms will be implemented. The testbed description section discusses how those classes and interfaces are used.

5. The Framework Testbed

In Section 2, we have presented some of the existing testbeds used to evaluate trust models, we have presented their features, advantages, and limitations. Our contribution is a new testbed that takes a step forward to provide a new way to evaluate trust models, and provide some new features that do not exist in the existing testbeds. Our testbed is based on an idea of a group of students. Each student is qualified in one or more topics. For example, we can find a student who is qualified in arithmetic operations such as addition or multiplication and also qualified in some physics operations. Each student has a set of homeworks to do, but he is not qualified in all of those homeworks. To simplify the processing in our testbed, we ignore homeworks that could be accomplished by the student itself and we just focus on homeworks that do not fit their skills. In this situation, an initiator student will seek help from his peers. Students will respond by a refuse or a proposal as a feedback for the request of the initiator student. Then, the initiator will use his history and received proposals and apply the trust metric strategy which present its social intelligence, then generate an evaluation of trustworthiness of each of his peers; such process will help the initiator choose the right student that will process the homework. Here, the right student does not mean the one who will certainly satisfy the initiator, but the student who is more likely qualified to satisfy the initiator from its point of view. The decision takes into consideration the knowledge experience that he acquires and the trust metric strategy that he is using to evaluate each of his peers. Each student in this group is simulated as an agent in the testbed, and he can play the role of a requester or a proposer or both of them.

5.1 Testbed architecture

As presented before, our testbed is based on the idea of the group students who try helping each other resolving their homeworks. There are two principal categories of students that we can distinguish: honest students and dishonest students. An honest student represents a student that has no intention to trick his peers, but it may sometime fail achieving some tasks with a specific probability. A dishonest student is a student with a bad intention while communicating with his peers. Every dishonest student follows an attack strategy. These attack strategies against trust and reputation systems differ in their natures and complexities from one to another. Some attacks are used against reputation systems where agents would ask for recommendation from other agents in the system such as Constant and Sybil attack [13]. Others are used to directly attack agents that acquired some positive/negative experience with the attacker agent [10]. We configure our testbed to handle the four known attacks: Constant or always dishonest attack, Camouflage attack, and Whitewashing attack. An agent that applies a constant dishonest attack always acts unfairly. This means that at each time the agent is selected by an initiator agent to perform a task, he will return a result different than what is expected by the initiator. This is the simplest attack because there is no timing strategy to trick peers over time or changing its identity. The second strategy that we introduce is the camouflage strategy. The idea of this attack is that the attacker gives fair result at the beginning of building its experience with his peers. Such behavior will initially give him a good reputation inside the system. Then, he will alter his behavior to trick the initiator. The third attack strategy introduced in our testbed is Whitewashing attack. An attacker agent who applies this strategy will leave and rejoin the system each time that he builds a bad reputation with his peers. The fourth attack is the random attack, where the agent randomly decides to respond fairly or unfairly.

Recall that the purpose of evaluating a trust model is to check whether or not the model has the capability of assessing the trustworthiness for each agent. The process of evaluating computational trust model refers to statistical measure classifications. In statistics, we find methods that help evaluating the quality of a model prediction [20]. Several works have been done to compare such methods and to show their consistency and efficiency within different scenarios [16]-[19]-[17]. Jurman shows that the Matthews Correlation Coefficient (MCC) method is the best suited method for evaluating the quality of binary classifications [26], where we have to classify our dataset into two groups. Trust evaluation models are an example of a binary classification where agents are grouped by their trustworthiness. In case that the trust evaluation values belong to a range, we can first convert them to match a binary repartition, and then use measure classifiers to evaluate the model.

Fig.4 illustrates the high level architecture of our testbed. The attack strategies and trust evaluation metrics are

configured to be used by the students who are represented by agents in the figure. The user will configure a test case by setting the number of honest students, the number of dishonest students, and will also specify the nature of each one of them. Later, the user defines the trust evaluation metrics that each agent requestor will use to evaluate his peers. When a test case starts, the monitoring dashboard could be used to preview the trust assessment values that set each of the task requester agent about his peers. Model evaluation metrics are used to evaluate the quality of the prediction of the trust model being used. Our testbed provide some features that are not available in existing testbeds.

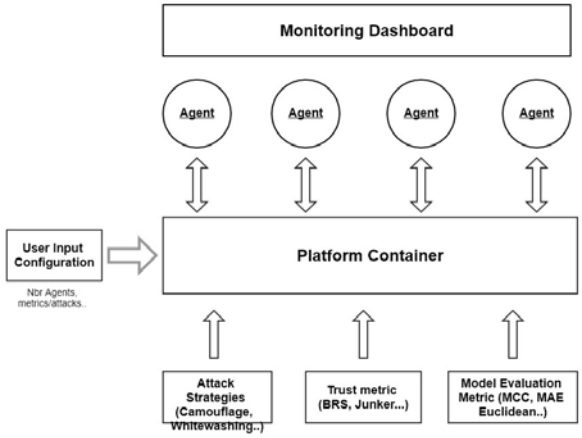

Fig.4 Testbed architecture

It can evaluates more than one trust models at the same time in the same test case. It can also evaluate those models against different attack strategies. Table 1 compares the characteristics of our testbed and some of the existing ones

Table 1 : Comparison of GeFMMAT testbed and existing ones

Testbed	Parallel evaluation	Attacks strategies	attacks Extensibility
Our testbed	Yes	Constant attack, Random attack White washing, Camouflage	Yes
ART	No	Random dishonest	No
TREET	No	Random dishonest	Yes
Lizi Zhang et al.	No	Constant attack, Sybil attack, Camouflage, Composite attack	Yes

Testbed	Model evaluation metric	System architecture	Different services
Our testbed	Evaluates the quality of prediction using correlation functions	decentralized	Yes
	MCC		
ART	evaluates the accuracy and cooperation achievable by the system of appraiser agents	decentralized	No
TREET	-the ratio of sales (profits) between honest and cheating sellers	centralized decentralized	No
Lizi Zhang et al.	Specific function proposed by authors	decentralized	No

5.2 Implementation and experimentation

The testbed was designed following the best practices applied in object oriented programming such as separation of concerns and design patterns. The importance of design patterns is that they represent proven solutions to commonly occurring problems in software design. They also help developers to describe and understand design solutions using well-known conventions. We will not discuss the implementation details of each component here; we will just give an overview of the design structure and the role of each part. The source code of our testbed is available online [27]. Our testbed is composed of five distinct components: behaviors, operations, tasks, features and trust models. Each part is implemented in a separated package.

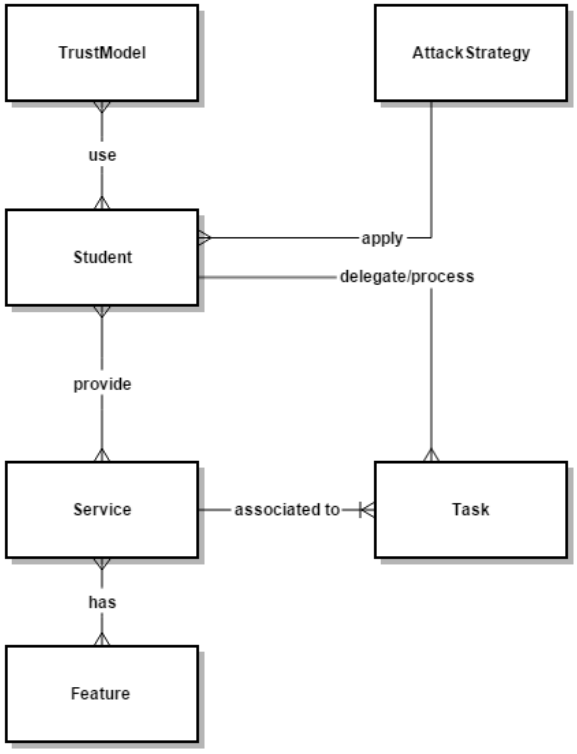

Fig.5 Testbed components

Fig.5 shows the relationships between those components. The package of students contains available behaviors that could be used by students. Those behaviors present the possible attack strategies. A student could play the role of a service requestor or a service provider or both of them at the same time. Roles played by a student are independent from his behavior nature; e.g., whether he is honest or dishonest. The operation package groups operations provided by students as services. Operations are independent from each others, and each operation has a set of features. For example, the addition operation has one feature, which is MathFeature; the CalculateSpeed operation has MathFeature and PhysicsFeature.

Each student could provide a set of operations to serve his peers. The default implementation of an operation presents the right implementation that results in correct answer, or let's say the honest implementation. A student who provides an operation may alter the result depending on the nature of his behavior. Honest students will use this implementation without any changes. Dishonest students alter the logic applied in those services. Each dishonest student uses his own alteration strategy. Each of the available operations is intended to process specific tasks. To this end, a distinct task model is defined for each operation. The package task groups task models. To evaluate the result and manage the trustworthiness of an agent's peers, the agent needs a trust metric, which is the implementation of a trust model. Those trust models are intended to be evaluated by the testbed. We have implemented four different trust models: Jonker trust model [14], Beta Reputation System (BRS) [18], Forgive Factor model [15], and an empty trust model called NoModel. It is also possible to introduce new models by implementing the interface ITrustMetric. The communication between students could be monitored using a sniffer tool provided by JADE platform. Fig.6 shows the screenshot of the interaction result between the set of students.

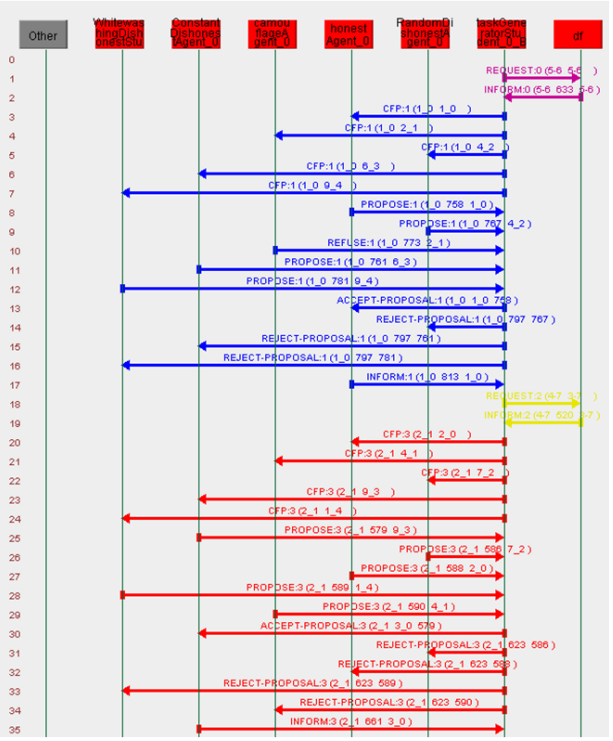

Fig.6 Interactions between agents

As shown in Fig.6, an initiator agent called taskGeneratorStudent_0_B first requests the list of registered students in the Directory Facilitator; then, it sends call for proposals to all of those students. Each student could respond with a proposal or a refuse. The initiator agent handles those responses and selects the best proposal depending on his trust knowledge. The selected agent will receive an accept proposal with the task to be performed, the rest will receive a reject proposal. When the selected agent performs the delegated task, it will send a result back to the initiator. The initiator will verify the validity of the result and updates its trust knowledge about the selected agent.

To evaluate the prediction quality and the robustness of implemented trust models against dishonest agents, the user needs to setup a configuration for an evaluation scenario. The configuration used to experiment the testbed in this paper is presented in Table 2.

Table 2 : An example of test case configuration

Category	Instance Number
Honest agents	5
Camouflage agents	5
Random dishonest agents	5
Always dishonest agents	5
Whitewashing agents	5
Task requester agents	4

It is possible to execute one or many task requester agents that will send a call for proposals to delegate their tasks. The testbed gives the possibility to evaluate more than one trust models in the same test case. This is done by configuring a task requester agent for each trust model. This evaluation scenario uses four task requester agents to evaluate the four implemented trust models at the same time. We have launched two test cases using this configuration. In the first test case, each task requester agent prepares and delegates 25 tasks, which is the number of task handler agents. In this case, the results will show the initial acquired trust assessment obtained using each of configured trust models. In the second test case, each task requester agent prepares and delegates 500 tasks. This case will show the convergence of trust assessments of those trust models. Results are presented in tables 3 and 4 that show the trust knowledge of a task requester about their peers grouped by category. The value of trustworthiness associated to an agent is in the form of belief/disbelief/uncertainty (selection time). The belief factor presents the degree of how an agent thinks that its peer is trustworthy. The disbelief factor presents the degree of untrustworthy that the agent thinks about his peer, and the uncertainty value presents the doubt that an agent still has about his peer. The selection time parameter shows how many times the agent has selected a peer of the category. Values within Tables 3 and 4 present means of evaluation values acquired by trust models about each category of students. For example, if the agent uses BRS as a trust metric, and delegates some tasks to five agents with the camouflage behavior (seven tasks in the case of 3), then the value shown in the table presents the mean evaluation of those five camouflage.

Table 3 : Case 1: 25 Tasks per requester

Trust model	Honest agent	Camouflage	Random
Junker	0,200/0,000 /0,800(6)	0,150/0,00 /0,850(5)	0,200/0,050 /0,750(5)
BRS	0,387/0,000 /0,613(10)	0,320/0,000 /0,680(7)	0,320/0,000 /0,680(7)
Forgive Factor	0,330/0,000 /0,670(4)	0,480/0,000 /0,520(11)	0,000/0,480 /0,520(4)
No model	0,000/0,000 /1,000(3)	0,000/0,000 /1,000(5)	0,000/0,000 /1,000(8)

Trust model	Constant Dishonest	Whitewashing
Junker	0,000/0,167 /0,833(3)	0,100/0,150 /0,750(6)
BRS	0,000/0,167 /0,833(3)	0,000/0,167 /0,833(3)
Forgive Factor	0,000/0,480 /0,520(4)	0,240/0,120 /0,640(3)
No model	0,000/0,000 /1,000(5)	0,000/0,000 /1,000(4)

Table 4 : Case 2: 500 Tasks per requester

Trust model	Honest agent	Camouflage	Random
Junker	0,900/0,100 /0,000(306)	0,000/1,000 /0,000(46)	0,150/0,750 /0,150(75)
BRS	0,922/0,050 /0,028(428)	0,244/0,404 /0,351(23)	0,302/0,369 /0,329(23)
Forgive Factor	0,371/0,229 /0,400(437)	0,255/0,345 /0,400(21)	0,171/0,429 /0,400(17)
No model	,000/0,000 /1,000(109)	00,000/0,000 /1,000(124)	0,000/0,000 /1,000(85)

Trust model	Constant Dishonest	Whitewashing
Junker	0,000/1,000 /0,000(20)	0,200/0,800 /0,000(53)
BRS	0,302/0,369 /0,329(23)	0,213/0,393 /0,393(16)
Forgive Factor	0,000/0,600 /0,400(7)	0,181/0,419 /0,400(18)
No model	0,000/0,000 /1,000(93)	0,000/0,000 /1,000(89)

Values used in Table 3 and Table 4 do not show the prediction quality of the used trust models. They just give a general idea about the defense of trust models against different attacks. To infer the prediction quality of each trust models, the results obtained by trust metrics (trust knowledge acquired during the task delegation process) need to be analyzed using measures discussed in Section 5 to extract the correlation between the prediction trustworthiness calculated by metrics, and the real trustworthiness. The MCC measure is used for this purpose. Table 5 shows the results obtained using those measures, and the number of satisfactions and dissatisfactions.

Table 5 : Trust model's correlation coefficient

Trust model	MCC	Satisfaction	Dissatisfaction
Junker	0.81	388	112
BRS	0,885	416	84
Forgive Factor	0,994	461	39
No model	0	218	282

MCC values are between -1 and 1; when its value is close to 1, it means that the metric is positively correlated with the expected values, which are the trustworthiness of agents in the system, and therefore, it has a good prediction quality. If its value is close to -1, it means that the metric is negatively correlated with the expected values. For small values close to 0, they means that there is no correlation between the results obtained while using the metric and the expected results.

6. Conclusion

In this paper we presented a JADE based testbed for evaluating learning techniques proposed by computational trust models. This testbed helps users compare the performance and efficiency of learning techniques proposed by existing trust and reputation models that uses probabilistic methods and machines learning techniques to evaluate trustworthiness of agents. In contrast to most of the existing testbeds, our testbed provides more flexibility while comparing those models within the system where agents manage and request different services. It provides also the possibility of testing many trust models at the same time in the same test case. For future work, we will study the impact of combination of different attacks strategies used to compromise the decision of an initiator agent. We will also try to introduce new possible attacks to view their impacts on the trust models used in this study. We also plan to improve the design of our testbed to handle and evaluate new trust models at runtime level.

References

[1] Liu, X., Trédan, G. ,& Datta, A., (2014). A generic trust framework for large-scale open systems using machine learning. Computational Intelligence , 30(4), 700-721.

[2] Pinyol, I., & Sabater, J. (2013). Computational trust and reputation models for open multi- agent systems: a review. Artificial Intelligence Review. 40, 1-25.

[3] Mifrah, Y., En-Nouaary, A., & Dahchour, M. (2016). An Abstract Framework for Introducing Computational Trust Models in JADE-Based Multi-Agent Systems. In Advances in Ubiquitous Networking (pp. 513-523). Springer Singapore.

[4] Fullam, K. K., Klos, T. B., Muller, G., Sabater, J., Schlosser, A., Topol, Z., Voss, M. (2005, July). A specification of the agent reputation and trust (art) testbed: experimentation and competition for trust in agent societies. In Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (pp. 512-518). ACM.

[5] Yu, B., & Singh, M. P. (2002, July). An evidential model of distributed reputation management. In Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1 (pp. 294-301). ACM.

[6] Youssef, M., Abdeslam, E. N., & Mohamed, D. (2015, October). A JADE based testbed for evaluating computational trust models. In 2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA) (pp. 1-7). IEEE.

[7] Kerr, R., & Cohen, R. (2010). TREET: The trust and reputation experimentation and evaluation testbed. Electronic Commerce Research, 10(3-4), 271-290.

[8] Jelenc, D., Hermoso, R., Sabater-Mir, J., & Trček, D. (2013). Decision making matters: A better way to evaluate trust models. Knowledge-Based Systems,52, 147-164.

[9] Jelenc, D., Hermoso, R., Ossowski, S., & Trcek, D. (2012). Alpha test-bed: A new approach for evaluating trust models. Infrastructures And Tools For Multiagent Systems, 51.

[10] Zhang, L., Jiang, S., Zhang, J., & Ng, W. K. (2012, May). Robustness of trust models and combinations for handling unfair ratings. In IFIP International Conference on Trust Management (pp. 36-51). Springer Berlin Heidelberg.

[11] The Foundation for Intelligent Physical Agents, http://fipa.org. 2002.

[12] Tryllian's Agent Development Kit, http://www.tryllian.com/adk.html.

[13] Al-Mutaz, M., Malott, L., & Chellappan, S. (2014). Detecting Sybil attacks in vehicular networks. Journal of Trust Management, 1(1), 1-19.

[14] Jonker, C. M., & Treur, J. (1999, June). Formal analysis of models for the dynamics of trust based on experiences. In European Workshop on Modelling Autonomous Agents in a Multi-Agent World (pp. 221-231). Springer Berlin Heidelberg.

[15] Burete, R., Bădică, A., & Bădică, C. (2010, July). Reputation model with forgiveness factor for semi-competitive E-business agent societies. InInternational Conference on Networked Digital Technologies (pp. 402-416). Springer Berlin Heidelberg.

[16] Hernández-Orallo, J., Flach, P., & Ferri, C. (2012). A unified view of performance metrics: Translating threshold choice into expected classification loss. Journal of Machine Learning Research, 13(Oct), 2813-2869.

[17] Huang, J., & Ling, C. X. (2007, January). Constructing New and Better Evaluation Measures for Machine Learning. In IJCAI (pp. 859-864).

[18] Commerce, B. E., Jøsang, A., & Ismail, R. (2002). The beta reputation system. In In Proceedings of the 15th Bled Electronic Commerce Conference.

[19] Parker, C. (2011, December). An analysis of performance measures for binary classifiers. In 2011 IEEE 11th International Conference on Data Mining (pp. 517-526). IEEE.

[20] Kirn, S., Herzog, O., Lockemann, P., & Spaniol, O. (Eds.). (2006). Multiagent engineering: theory and applications in enterprises. Springer Science & Business Media.

[21] Sabater, J., & Sierra, C. (2001). Social regret, a reputation model based on social relations. ACM SIGecom Exchanges, 3(1), 44-56.

[22] Tofallis, C. (2015). A better measure of relative prediction accuracy for model selection and model estimation. Journal of the Operational Research Society,66(8), 1352-1362.

[23] Pokahr, A., & Braubach, L. (2009). A survey of agent-oriented development tools. In Multi-Agent Programming: (pp. 289-329). Springer US.

[24] http://soar.eecs.umich.edu/

[25] The Juxtapose Project, https://jxta.kenai.com/

[26] Jurman, G., Riccadonna, S., & Furlanello, C. (2012). A comparison of MCC and CEN error measures in multi-class prediction. PloS one, 7(8), e41882.

[27] https://github.com/mifmif/JADETrustTestbed