Towards a Framework for the Analysis and Evaluation of Computational Trust Models in Multi-agent Systems

Mifrah Youssef *, En-Nouaary Abdeslam, Dahchour Mohamed
Institut National des Postes et Télécommunications, Rabat, Morocco.

* Corresponding author. Tel.: 00212634653721; email: mifrah@inpt.ac.ma, abdeslam@inpt.ac.ma, dahchour@inpt.ac.ma
Manuscript submitted August 25, 2017; accepted October 13, 2017.
doi: 10.17706/jsw.12.11.892-905

Abstract: 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 have been established to evaluate trust and reputation models and verify their robustness and efficiency. However, a lack of flexibility to handle scenarios related to multi-context trust models arise with those testbeds. We present in this paper a framework for evaluating computational trust models that provides to users more flexibility while comparing trust models in open systems and shows analysis results in chart diagrams. The ultimate objective is to evaluate and classify available computational trust models.

Key words: Computational trust, agent-oriented software engineering, reputation, testbed, agent simulation, 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, as defined by Wooldrige [1], are a kind of systems that consist of a set of agents who interact with each other to perform complex tasks. 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 good examples 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 computational trust model into MAS, agents that belong to this system become able to rate their partners on the basis of their interaction results. Researchers in this field have proposed various computational trust models [2], [3], [4], [5]. A detailed classification have been established by Pinyol and Sabater [6] illustrate some of the existing computational trust models classified by a set of dimensions such as information sources, model type and visibility. While designing an open MAS, and without a prior knowledge about differences between existing trust models, the designer that try to introduce a trust model to the system would probably be confused about which one will fit best. The experiments of computational trust models proposed by their authors are not always enough, especially when non experimented malicious behaviors should be considered. To this end, research community have designed comparative tools called testbeds used for testing the efficiency of computational trust models within configured scenarios. The advantage of the testbed concept is the ability to test two computational trust models with the same test case configuration. But the lack in existing testbeds is that trust in not presented as a multi-context concept. Another missing feature in the majority of existing testbeds is the limited set of attack strategies against evaluated trust models, and the ability to extend to new ones. To this end, we propose a testbed that presents two advantages: on the one hand, the proposed testbed manages trust assessment per context by introducing and managing diverse services, and on the other hand the robustness and efficiency of computational trust models are evaluated against different malicious behaviors which are implemented as attack strategies.

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

2. Trust and Reputation Testbeds: A Systematic Overview

Trust and reputation testbeds are intended for the evaluation and comparison of computational trust and reputation models in various situations. The design of such tools requires a global view of existing trust and reputation models, and a formalization of a global configurable scenario used in generated test cases. The first and most known testbed is The Agent Reputation and Trust Testbed (ART) [4]. This testbed provide an environment of competing agents. Each agent have limited expertise in evaluating paintings. An agent will gain a utility by appraising the value of paintings as a response of a requester. An agent can also respond to a reputation request where the requester ask for reputation information about a third agent. Computational trust and reputation models are used here to improve the decision making process and eventually increase the agent utility. In ART, attack strategies that malicious agents could use are not detailed; they consider malicious agent as an agent that will exchange non credible information. Kerr and Cohen [7] were inspired by ART testbed, and they create a, extended testbed, called The Trust and Reputation Experimentation and Evaluation Testbed (TREET) [7]. A marketplace simulation scenario was used In TREET, where buyers and sellers exchange goods. Sellers are motivated by the profit they make from sales, while buyers are motivated by the value of items. The behavior of malicious sellers is defined by

not shipping the item which will increase their profits. The advantage of TREET is one the one hand there is no constraint on trust component structure, thus, the user define the trust structure component from scratch. On the other hand, the user can implement new collision attacks via some provided utilities. But with multi-context computational trust model, TREET fail in giving efficient model evaluation. Another recent testbed proposed by Zhang [8] includes a set of attacks strategies that help verifying the robustness of evaluated trust models against such attacks. Whilst, it uses the scenario of buyers and sellers with a single service, this still suffers from a lack of flexibility and could cause loss of information in case of multi-context computational trust model. Jelen in his study analyze the influence of the decision making mechanism on trust models, he proposes a The Alpha Testbed (ATB) that try to prove his hypothesis related to the impact of decision making on a trust model ,and concludes that the decision making mechanisms influences the performance of trust models [9]. The current status of existing testbeds arise a challenge of designing a new testbed that groups a set of features and add new ones, such as the possibility of managing more than one service in parallel to handle multi-context computational trust model, and evaluate trust models against different attack strategies with the possibility to extends or adds new ones.

3. GEFMMAT Computational Trust Framework

The research community of MAS knows a fast development, due to the diversity of application domains of such systems, like healthcare [10], energy systems [11] and transportation and simulation [12]. Researchers and designers have proposed various models and frameworks to design MAS. Each of the proposed frameworks uses a specific domain model that set-up agent systems with respects to specific perspective. There are models that treats MAS from a macro-level, where they focus more on organizational and hierarchical aspect [13], and include notions related to this perspective such as environment, hierarchy and role. Another kind of MAS models try to handle the micro-level of MAS related to individual agent and the interaction aspect between agents in a system [14]. However, the proposed models and frameworks did not handle explicitly the concept of trust by defining components and workflows where this concept is intended to be in use. To introduce the concept of trust in MASs, we began with a listing of commons concepts usually found in open MAS. Afterwards, we start working on a generic framework based on those concepts. As a result, we have designed 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 MASs.

Meta-classes of our GeFMMAT meta-model presented in Fig.1 try to handle common concepts that could be found open dynamic MAS.

Each meta-class has one or more relationships with others. The pivotal *Agent* meta-class illustrates the autonomous agent which interacts with their peers by providing or asking for services. The defined Meta-class doesn't set any constraint on the internally specification of the agent model, thus, the designer could conceives customized features that will characterize agents within the system. Agent's features would change from a system to another. In an electronic commerce platform For example, an agent would have a profile composed of information such as its user name, country, registration time and experiences. The expertise that an agent can propose to other agents is presented by the meta-class *Service*. An agent can provide one or more services to perform tasks delegated by their peers. A service can have at least one activity context which defines a specific area of expertise. The *Communication Knowledge* forms a communication protocol definition used during the communication process between agents. This protocol defines how an agreement between two agents will be confirmed, what requirements should be shared before performing the task, and how results will be received. A task that an agent exchange and delegate is modeled by The *Task* meta-class. Each task is related to an activity context that present its application

context. When an agent delegate a task to its partner, the agent initiator will have a *Relationship* entity that groups information related to its relationship with his partner. This relationship is defined as a directional relation between an initiator agent and its partners, and it contains passed experiences of the initiator with his partner, the trust assessment values that the initiator set to its partners, and also the computational trust metrics used to evaluate. The partner agent send task result to the initiator when he finishes task processing, the initiator will use this result with other parameters and apply the trust metric to assess his partner's trustworthiness. Next, he updates his trust knowledge. The current state of the art about computational trust models shows that there is a variety of data structure presentation of trust proposed by researchers. Some presentation of the structure of trust component are defined as a variable that takes a value from a set of finite value space. Others define the values space as a range of numeric. Another more generic definition is to define the trust component as a vector in a multidimensional space where we can set to each agent's feature an evaluation value [14] or defining trust assessment using subjective logic [15]. The subjective logic format of trust helps defining state where trust information is unavailable such as for newcomer agent due to the integration of an uncertainty dimension. Each computational trust model provide a process to be applied to assess and update trust value, such process is defined as a trust metric , and it is presented in our meta-model via The Trust Metric meta-class. An agent could associate to each of their partners a trust metric, this means that an agent could uses more than one trust metrics. Agents that share specific features are associated to a group, thus, belonging to a group is helpful while identifying the nature of an agent. The designer could statically defines a group while designing the system, or implement a process to be executed at runtime level which will define a group dynamically. A group presents a form of aggregation that would help agents evaluating the trustworthiness of group's members such as when the trust assessment of an agent is unavailable. The meta-model presents the framework's components structure. The framework defines a generic workflow to be applied by agents while delegating their tasks or requesting for information, in this workflow, a sequence of steps are applied by agents to improve the decision making process. The process starts by a call for proposals where an agent initiator ask for proposals from the environment for his task, then it continue with steps that shows how agents should manage information about the environment, and how to use feedbacks after the decision process. It defines where an agent will use the trust model metric, and when its trust knowledge will be updated. Fig. 2 presents the framework workflow for the management of the trust.

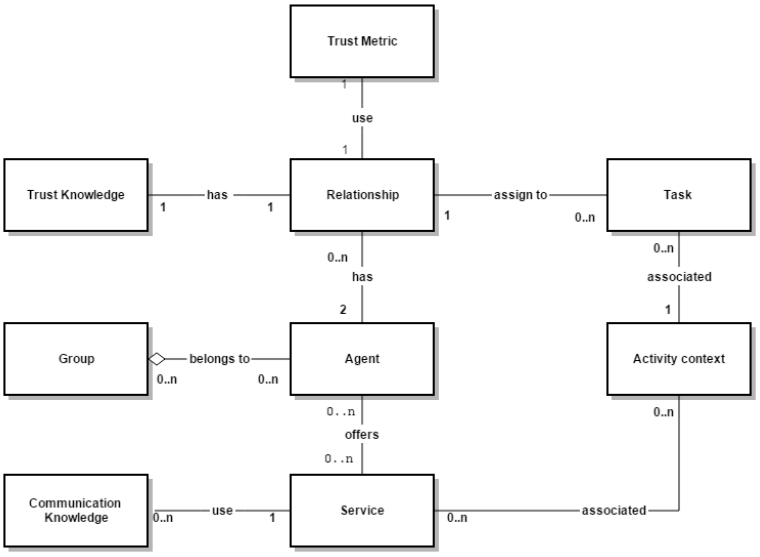


Fig. 1. Meta-model of the GeFMMAT framework.

4. GEFMMAT Implementation in JADE

Over the last few years, the research community in MAS field has established practical development platforms dedicated to MAS programming and simulation. Some MAS platforms are oriented middleware and helps implementing interoperable agent systems, such as Java Agent DEvelopment Framework (JADE) [16] and Agent Development Kit (ADK) [17]. 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 SOAR [18] and JASON [19]. Detailed classifications have been established using various dimension and evaluation criteria to compare Agent platforms existed in the literature [2], [20], [21]. Each platform is based on a set of standards and specifications such as FIPA [22], JXTA [23], and web services. JADE Platform adopt FIPA specification for interoperable intelligent MASs. It uses also an agent abstraction to design agent in the system. JADE platform includes graphical tools useful in the development lifecycle. With the advantage that offer JADE over the other platforms, we have selected JADE as a platform for the implementation of our Framework. Fig.3 shows the class diagram of the GeFMMAT meta-model implemented in JADE.

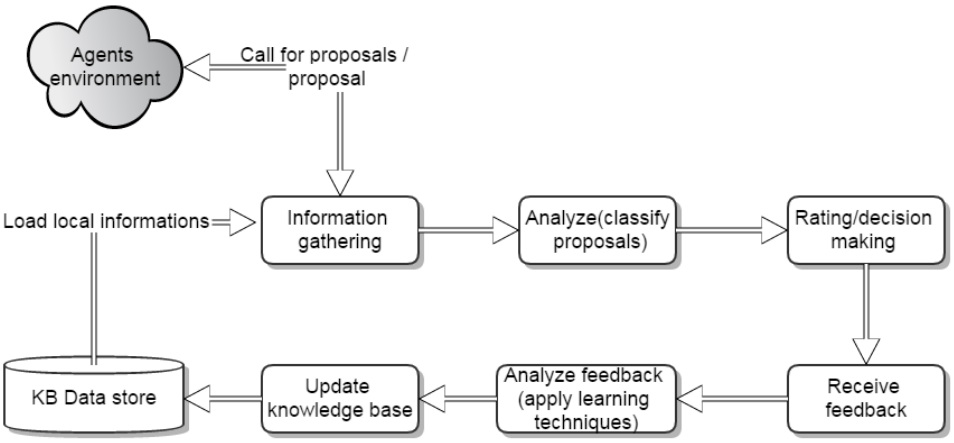


Fig. 2. GeFMMAT Workflow.

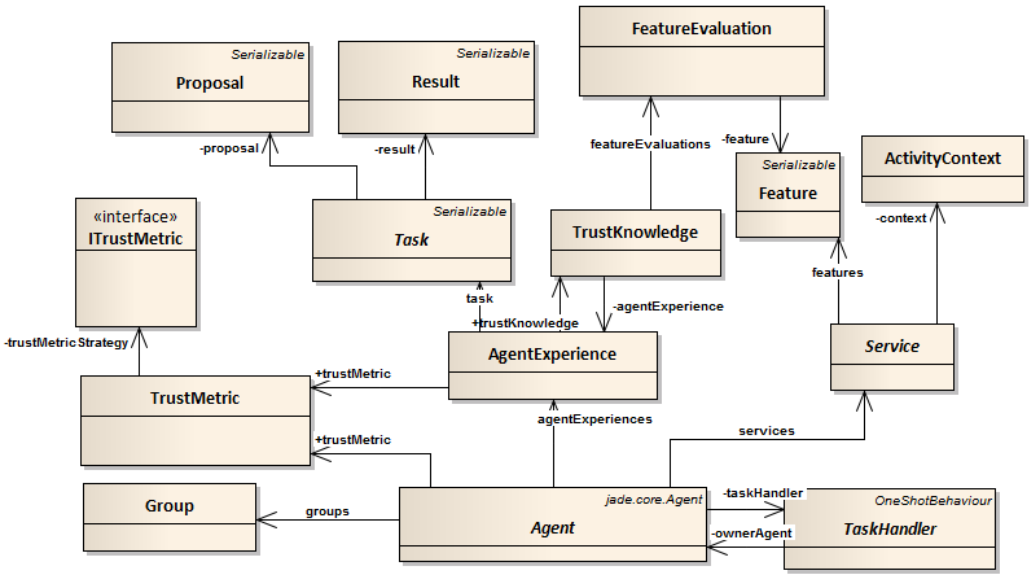


Fig. 3. Implementation of GeFMMAT meta-model in JADE.

5. An Evaluating Testbed

In our brief survey presented in Section 2, we have described some of the existing trust and reputation testbeds, their features, advantages, and limitations. Our work focus on designing a featured testbed that provides a new way to evaluate trust models, with new features that didn't exist in previous testbeds. In our testbed, we defines a group of agents that play the role of students, where each student have qualification in one or more skills that presents services provided by this agent. The set of available skills is predefined in the testbed, for an example a student can have a qualification in arithmetic operations such as addition or multiplication or in some physics operations. A student has a set of tasks to do in the form of homeworks. An initiator student will send a call for proposals to their peers to ask them for providing service proposals. Each service provider agent will respond to the request of the initiator by a refuse or a proposal. Then, the initiator will use the received proposals that contains a price for the service and his trust knowledge history to choose the best peer; Here, the best student does not necessarily 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 with a suitable price. This decision takes into consideration the computational trust metric used to evaluate the student and the past experiences. A student in the system can plays the role of services provider or services requester or both of them at the same time.

5.1. Overall Architecture

This section describes the architecture of our evaluating testbed. The agents' community presented in testbed scenarios is a set of students that provides or request for services. Service providers students can be divided into two different groups: honest students and dishonest students. An honest student represents a student whose behavior and intention matches their proposals. On the contrary to honest students, a dishonest student hides bad intention while exchanging with their peers. A malicious behavior of a dishonest student may follows a pattern that is presented as an attack strategy. There are several attacks strategies against trust and reputation systems that vary in their natures and complexities from one to another. Some attacks are used to directly attack agents that acquired some positive/negative experience with the attacker agent [8], while others are used against reputation systems where agents would ask for recommendation from other agents in the system such as Constant and Sybil attack [24]. Our testbed handle four known attacks related to computational trust systems: Camouflage attack, Random attack, Constant (or always) dishonest, and Whitewashing attack. The Constant dishonest attack presents a behavior that never satisfy the requester. This attack is a sample attack because it doesn't need to process requests or earn initial reputation. Another simple attack is the random strategy where the attacker randomly decides to respond fairly or unfairly. With the camouflage strategy, the attacker tries to give fair result at the beginning of building its reputation experience with his peers. Such behavior will initially help the attacker earning good reputation. Then, he will alter his behavior to trick their peers. An attacker who follows the whitewashing attack will leave and rejoin the system each time his reputation decent to a certain level within the system.

The purpose of trust and reputation testbed as described in section 2, is to check whether or not the trust model has the capability of assessing trust values to agents and the reliability of this assessment. Evaluating a computational trust model may refers to statistical measure classifications which provide methods that shows the quality of a model prediction [25]. Several studies related to data mining and classification have been established to propose and compare various classifiers and to show their prediction quality and performance [26] [27], [28]. Jurman *et al*. [29] shows that the Matthews Correlation Coefficient (MCC) method gives reliable results about the quality of binary and multi-class prediction classifiers. Trust evaluation models are a kind of classifier where agents are classified based on their trustworthiness. Trust

value spaces can be subdivided to a finite set of ranges, and each range will represents a class of agents, we can then use a classifier based on this subdivision to evaluate the prediction quality of a trust model.

The Testbed has the set of attack strategies previously presented in this section, and the computational trust metrics that can be used by service requesters' agents. To start a test case scenario, user have to set a test case configuration composed by the numbers of honest and dishonest agents , attack strategies , and trust metric used by each agent. The architecture of our testbed is illustrated in Fig.4. , the monitoring dashboard is used to preview trust assessment applied by each of service requester agent. The Model evaluation metrics configuration defines classifiers that provide the testbed to evaluate the prediction quality of trust model metrics being used in a test case scenario.

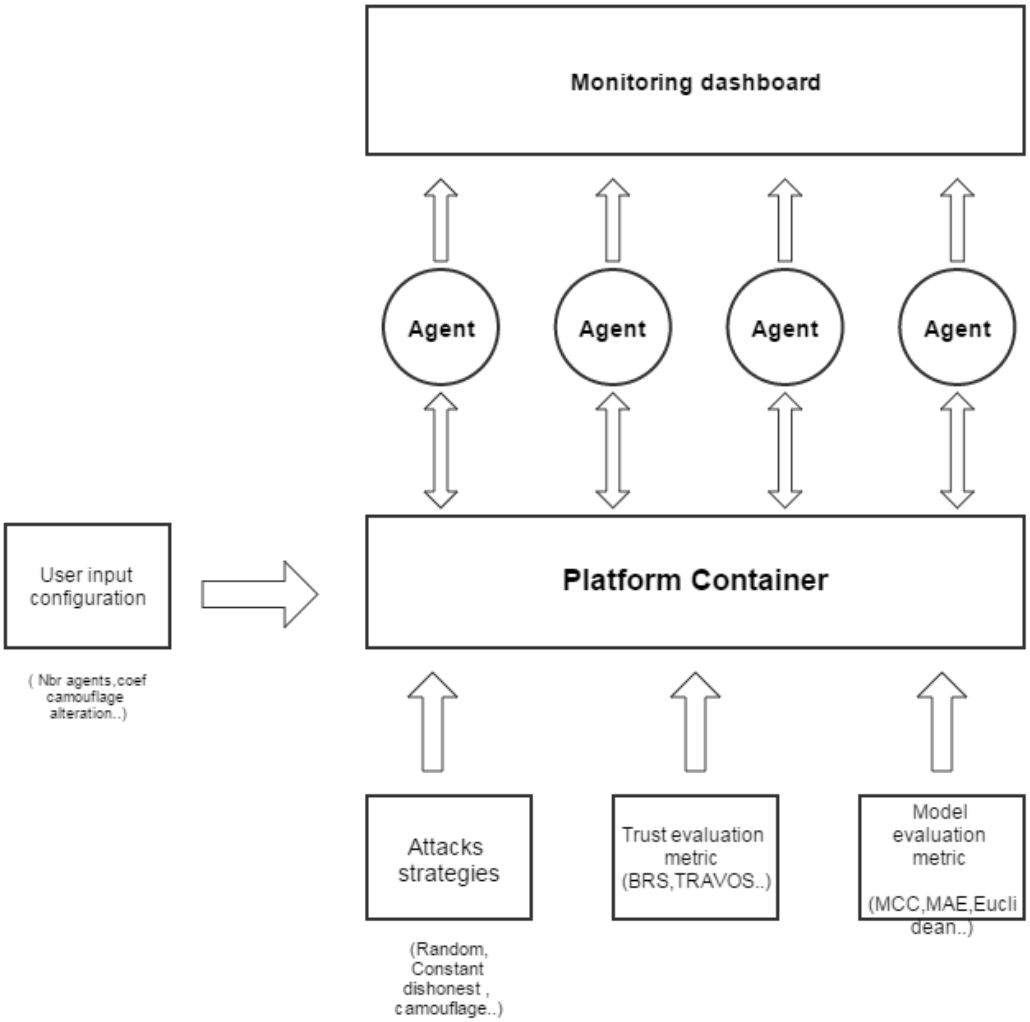


Fig. 4. Architecture of the testbed.

The user configuration of a test case scenario may include one or more trust models that would be evaluated in the same test case, and it can also evaluate those models against different attack strategies. Table 1 shows the characteristics of the existing testbed.

5.2. Testbed Implementation

In software design, concepts such as Separation of Concerns (SoC) and design patterns formalize best practices that simplifies the development and maintenance of software application. They define

relationships and interaction between classes or objects involved in commons situations. The importance of the general solutions provided by such practices appear when we maintains or introduces new features to the application. They provide also a self-description of the designed solution using well-known conventions shared amongst developers. We have followed those practices while designing and implementing our testbed .We are unable to discuss the implementation details of each component in this paper, but we will give an overview of the design structure and the role of each part. The implementation code of the testbed is available at our public repository [30] that contains also some of our previous published works [31], [32]. There are six distinct components that compose the testbed: Students, Operations, Attack strategies, Tasks, Features and Trust models, Fig.5 shows the relationships between those components. Each part is implemented in a separated package. Students package contains available students that exist within the. Attack strategies package groups the list of possible attack strategies. Operations package groups operations provided by students as services. Operations are independent from each other's, and each operation has a set of features from the Feature package.

Table 1. Our Testbed and Existing Ones

Testbed	**Parallel evaluation**	**Attacks strategies**	**Model evaluation metric**	**Extensible for new attacks**	**System architecture**	**Different services**
TREET	No	Random dishonest	-the ratio of sales (profits) between honest and cheating sellers	Yes	centralized decentralized	No
ART	No	Random dishonest	evaluates the accuracy and cooperation achievable by the system of appraiser agents	No	decentralized	No
Lizi Zhang et al.	No	Constant attack, Sybil attack, Camouflage, Composite attack	Specific function proposed by authors	Yes	decentralized	No
Our testbed	Yes	Constant attack, Random attack White washing, Camouflage	Evaluates the quality of prediction using correlation functions MCC	Yes	Decentralized	Yes

Service providers have a set of operations available to serve service requester. An Operation class is implemented by default following the honest behavior that results correct answers. Honest students will apply operation's implementation without any changes. When a service provider follows an attack strategy, the result of an operation may be altered based on the attack strategy followed. Each of the available operations is intended to process specific tasks. To this end, a distinct task class is defined for each operation class. Tasks package groups task models available within the system that a service requester student can prepare and ask for processing it. Trust models encapsulate trust metrics logics applied by service requester to evaluate trustworthiness of a peer after receiving the result. Those trust models metrics are intended to be evaluated by the testbed against the attack strategies.

5.3. Experiments and Results

To experiment our testbed, We have introduced implementations of three different trust models metrics: *Beta Reputation System* (BRS) [15], *Forgive Factor* [5] and *Jonker* [4]. We experiments also an empty trust model called *NoModel* that will shows us the decision process of service requesters without using any trust models applied. The objective of this experimentation is to evaluate the prediction quality and the robustness of implemented trust models against the attack strategies, we uses the configuration presented in Table 2 for our tests cases scenarios.

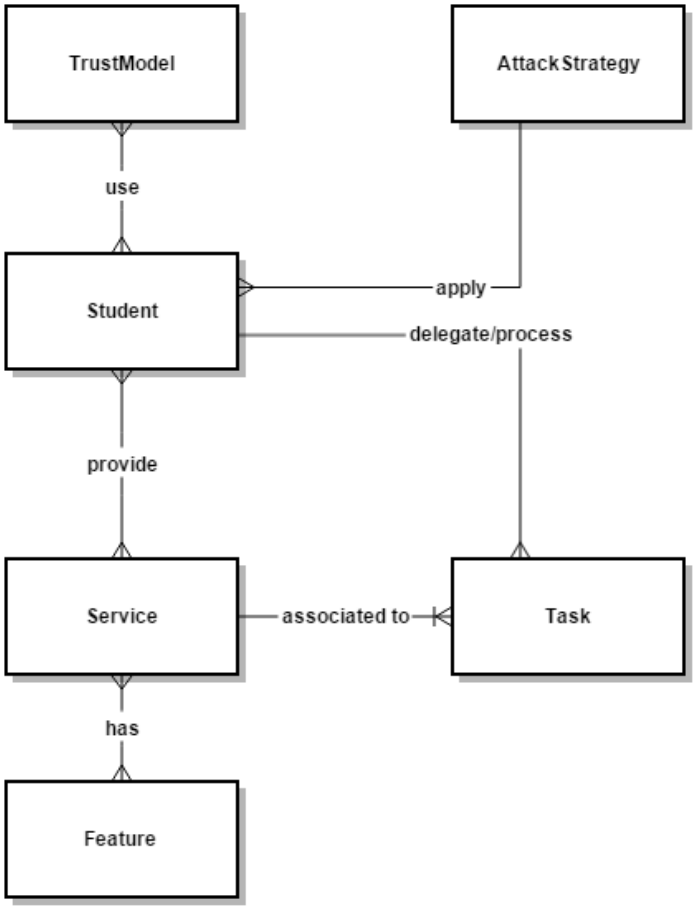


Fig. 5. Testbed components.

Table 2. Test Case Configuration

Category	**Instance Number**
Honest agent	4
Whitewashing agent	4
Random dishonest agent	4
Always dishonest agent	4
Camouflage agent	4
Task requester agent with BRS metric	1
Task requester agent with Junker metric	1
Task requester agent with Forgive Factor metric	1
Task requester agent with NoModel	1

As presented in the previous section, our Framework implementation is based on JADE platform. When we starts the test case scenario, we can monitor the communication between students using sniffer tool

provided by JADE platform. Fig.6 shows the interactions between configured agents while executing the test case scenario.

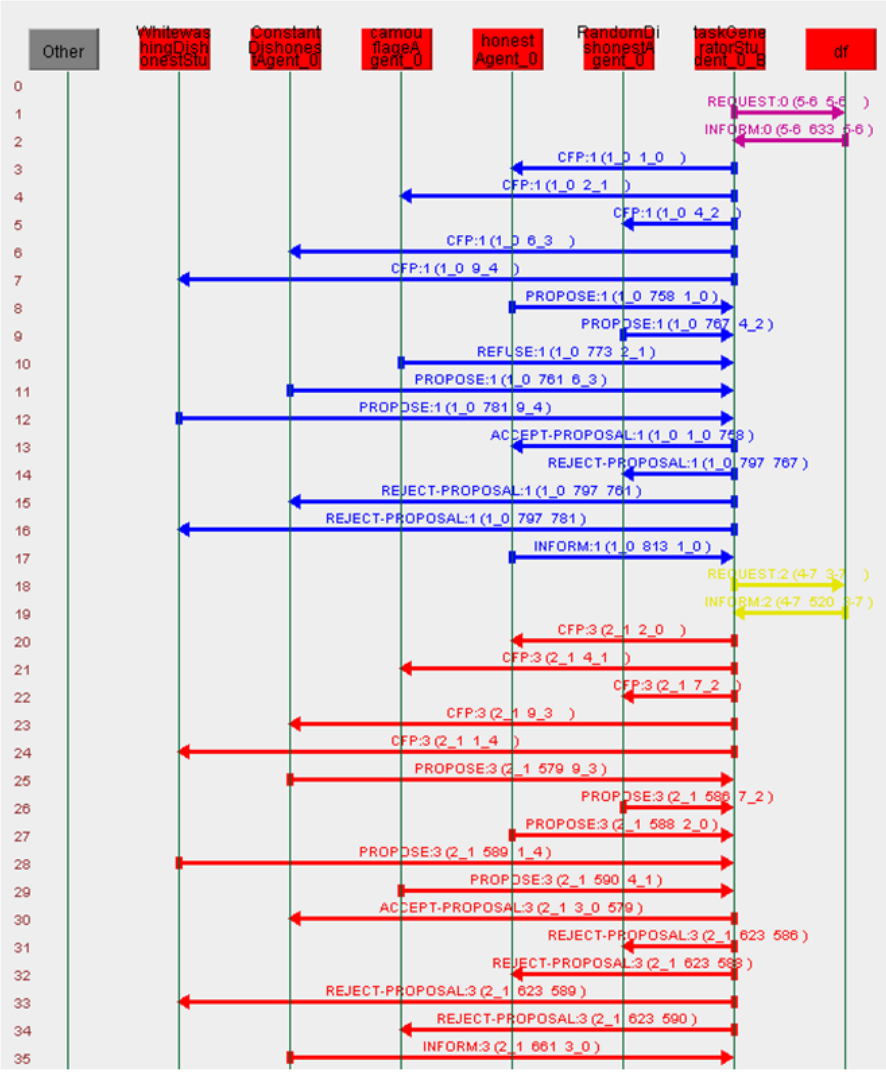


Fig. 6. Agents Interactions.

In Fig. 6, we observes an initiator agent called taskGeneratorStudent_0_B broadcast a call for proposals to the list of available service providers' agents. Each agent respond to this request with a proposal or a refuse. After receiving all responses from service providers' agents, the initiator agent analyzes proposals and use his trust knowledge to selects one of the proposer. The selected agent will receive an accept proposal with the task to be performed, others will receive a reject proposal. When the selected agent performs the delegated task, it send back the result to the initiator. The status of the result are used to updates the trust knowledge of the current service provider agent.

In this experiment configuration, we uses one task requester agent for each trust model to evaluate those trust models in the same test case. We have launched two test cases using this configuration. In the first test case, each task requester agent delegates a total of 25 tasks. This results in the status of the initial trust assessment obtained using each trust model. The second test case, we increases the number of tasks to be delegated by each agents to 500 tasks. The second case will provides a view on how trust assessments is updated using each of used trust models implemented. The Results of the two test cases are presented in Tables 3 and 4 shows the mean values of trust assessments affected by each task requester to their peers

grouped by attack strategies. The structure of trust value assigned to agents is adapted to the subjective logic format which is in the form of <belief>/<disbelief>/<uncertainty> (<selection time>). The belief dimension presents the degree of trustworthiness of an agent. The disbelief dimension presents the degree of untrustworthiness of the agent, and the uncertainty dimension presents the uncertainty about the behavior of that agent. We added a selection time dimension that shows the number of selection times an agent has.

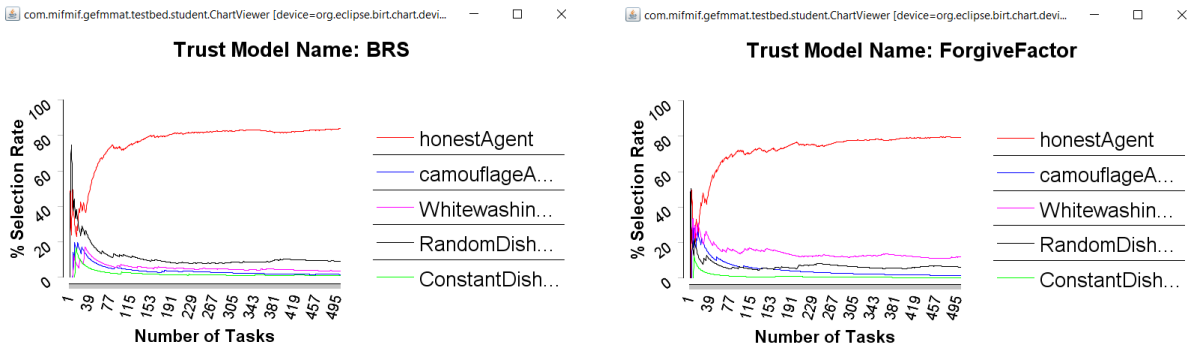
Table 3. Case 1: 25 Tasks Per Requester

Trust model	Honest agent	Camouflage	Whitewashing	Random	Constant Dishonest
Junker	0,150/0,000 /0,850(3)	0,300/0,000 /0,700(6)	0,150/0,100 /0,750(7)	0,200/0,100 /0,700(6)	0,000/0,150 /0,850(3)
BRS	0,510/0,000 /0,490(12)	0,200/0,000 /0,800(4)	0,000/0,067 /0,933(2)	0,000/0,083 /0,917(3)	0,000/0,200 /0,800(4)
Forgive Factor	0,240/0,000 /0,760(5)	0,160/0,000 /0,840(7)	0,330/0,080 /0,590(5)	0,240/0,240 /0,520(4)	0,000/0,300 /0,700(4)
No model	0,000/0,000 /1,000(2)	0,000/0,000 /1,000(8)	0,000/0,000 /1,000(3)	0,000/0,000 /1,000(6)	0,000/0,000 /1,000(6)

Table 4. Case 2: 500 Tasks Per Requester

Trust model	Honest agent	Camouflage	Whitewashing	Random	Constant Dishonest
Junker	0,750/0,100 /0,150(351)	0,000/0,950 /0,050(41)	0,250/0,550 /0,200(51)	0,150/0,600 /0,250(41)	0,000/0,800 /0,200(16)
BRS	0,904/0,069 /0,027(401)	0,320/0,427 /0,253(30)	0,391/0,295 /0,314(27)	0,361/0,383 /0,256(30)	0,000/0,520 /0,480(12)
Forgive Factor	0,448/0,152 /0,400(446)	0,237/0,363 /0,400(25)	0,104/0,496 /0,400(10)	0,120/0,480 /0,400(12)	0,000/0,570 /0,430(7)
No model	0,000/0,000 /1,000(93)	0,000/0,000 /1,000(103)	0,000/0,000 /1,000(110)	0,000/0,000 /1,000(99)	0,000/0,000 /1,000(95)

Values used in Table 3 and Table 4 shows in details the final result of the execution of a test case. They give a general idea about the status trust knowledge of an agent when using a trust model against each of the attacks strategies. The testbed includes a functionality of drawing visual charts that display selection rate of existing service providers based on their behavior nature, which provides a global view about the evolution of selection process using each of the evaluated trust metrics.



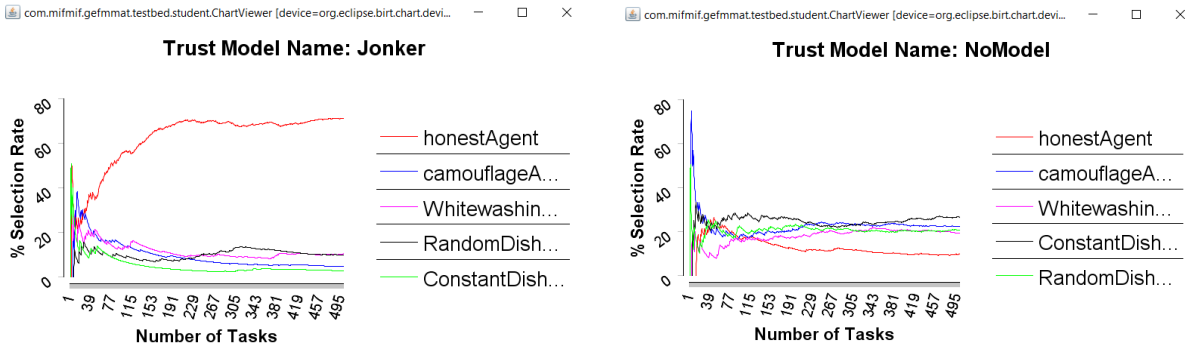


Fig. 7. Evolution of selection rate by number of tasks.

To measure the prediction quality of evaluated trust metrics, we analyses trust assessments values obtained by trust metrics using classifiers discussed in Section 5 to calculate the correlation between trustworthiness calculated by trust metrics, and the real behavior of the agent. As discussed in the previous section, we uses MCC as our trust metric classifier. Results obtained using MCC are presented in Table 5, we included two columns that present the number of satisfactions and dissatisfactions of each service requester during the test case of 500 tasks.

Table 5. Correlation Results of Trust Models

Trust model	MCC	Satisfaction	Dissatisfaction
Junker	0,814	393	107
Forgive Factor	0,874	419	81
BRS	0,95	454	46
No Model	0	210	290

The calculated correlation coefficient using MCC belongs to the range [-1, 1], this value reflect a high positive correlation when it approach to 1, and a high negative correlation when it approach to -1, when it is near to 0 it means that the trustworthiness calculated by trust metrics doesn't correlates with the real behavior of the agent such as the case of the agent with No Model. We can also see in the obtained results that agent with implemented trust models has MCC value near to 1 which proof their ability to classify agents based on their behaviors, for example BRS Trust Model has a high correlation coefficient, which reflect that using such trust model helps distinguishing trusted agents and increases the satisfaction rate of delegated tasks.

6. Conclusion

We have designed an evaluating Framework for computational trust models in MAS. Our Framework helps users comparing the efficiency and performance of existing trust models. The Framework provides advantages over existing testbeds in handling multi-context trust models, and it shows a flexibility while comparing those models within a system with various attacks strategies using visual presentation format of the results. It is also possible to evaluate more than one trust models in one test case. For future works, 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 Framework to handle and evaluate new trust models at runtime level.

References

[1] Wooldridge, M. (2009). *An Introduction to Multiagent Systems*. John Wiley and Sons.

[2] Bordini, R. H., Braubach, L., Dastani, M., Seghrouchni, A. E. F., Gomez-Sanz, J. J., Leite, J., & Ricci, A. (2006). A survey of programming languages and platforms for multi-agent systems. *Informatica*, *30(1)*.

[3] Yu, B., & Singh, M. P. (2002, July). An evidential model of distributed reputation management. *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1* (pp. 294-301).

[4] Jonker, C. M., & Treur, J. (1999, June). Formal analysis of models for the dynamics of trust based on experiences. *European Workshop on Modelling Autonomous Agents in a Multi-Agent World*.

[5] Burete, R., Bădică, A., & Bădică, C. (2010, July). Reputation model with forgiveness factor for semi-competitive E-business agent societies. *Proceedings of the International Conference on Networked Digital Technologies* (pp. 402-416).

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

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

[8] [8] Zhang, L., Jiang, S., Zhang, J., & Ng, W. K. (2012, May). Robustness of trust models and combinations for handling unfair ratings. *Proceedings of the IFIP International Conference on Trust Management* (pp. 36-51).

[9] 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.

[10] Shakshuki, E., & Reid, M. (2015). Multi-agent system applications in healthcare: Current technology and future roadmap. *Procedia Computer Science*, *52*, 252-261.

[11] Sun, Y., Tian, Y., & Xie, X. J. (2017). Stabilization of positive switched linear systems and its application in consensus of multi-agent systems. *IEEE Transactions on Automatic Control*.

[12] Horni, A., Nagel, K., & Axhausen, K. W. (Eds.). (2016). *The multi-agent transport simulation MATSim*. London: Ubiquity Press.

[13] Isern, D., Sánchez, D., & Moreno, A. (2011). Organizational structures supported by agent-oriented methodologies. *Journal of Systems and Software*, *84(2)*, 169-184.

[14] Yokoo, M. (2012). Distributed constraint satisfaction: foundations of cooperation in multi-agent systems.

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

[16] Bellifemine, F. L., Caire, G., & Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*. John Wiley & Sons.

[17] Tryllian's Agent Development Kit. Retrieved from http://www.tryllian.com/adk.html

[18] Eecs. Umich. Retrieved from http://soar.eecs.umich.edu/

[19] 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.

[20] Kravari, K., & Bassiliades, N. (2015). A survey of agent platforms. *Journal of Artificial Societies and Social Simulation*, *18*(*1*), 11.

[21] Pokahr, A., & Braubach, L. (2009). A survey of agent-oriented development tools. *Multi-Agent Programming.*

[22] The Foundation for Intelligent Physical Agents. Retrieved from http://fipa.org. 2002.

[23] The Juxtapose Project. Retrieved from https://jxta.kenai.com/

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

[25] Kirn, S., Herzog, O., Lockemann, P., & Spaniol, O. (2006). *Multiagent Engineering: Theory and Applications in Enterprises*. Springer Science and Business Media.

[26] 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.

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

[28] Huang, J., & Ling, C. X. (2007, January). Constructing new and better evaluation measures for machine learning.

[29] Jurman, G., Riccadonna, S., & Furlanello, C. (2012). A comparison of MCC and CEN error measures in multi-class prediction.

[30] Mifmif. Retrieved from https://github.com/mifmif/JADETrustTestbed

[31] Mifrah, Y., En-Nouaary, A., & Dahchour, M. (2016). An abstract framework for introducing computational trust models in JADE-based multi-agent systems. *Advances in Ubiquitous Networking* .

[32] Youssef, M., Abdeslam, E. N., & Mohamed, D. (2015, October). A JADE based testbed for evaluating computational trust models. *Proceedings of the 2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*

Youssef Mifrah received his engineer degree in software engineering and computer science from L'Institut National des Postes et Télécommunications (INPT) Rabat, Morocco, in 2012. He is currently a senior software developer and PhD student in the doctoral center of INPT. His current research topics are on modeling trust and reputation in multi-agents systems.

Abdeslam En-Nouaary received his engineer degree in computer engineering, option data communication and computer networks from ENSIAS (École Nationale Supérieure d'Informatique et d'Analyse des Systèmes), Rabat, Morocco, in 1996, the M. Sc. and Ph. D. degrees in computer science from the University of Montreal in 1998 and 2001 respectively. Dr. En-Nouaary is currently an associate professor at INPT (Institut National des Postes et Télécommunications), Rabat, Morocco. Before joining INPT in 2008, Dr. En-Nouaary has been an associate professor at the Electrical and Computer Engineering Department of Concordia University, Montreal, Canada, From 2001 to 2008. His main research interests are modeling and validation of distributed, realtime, and embedded systems.

Mohamed Dahchour received the doctoral (2001) in computer science from Ecole polytechnique de Louvain, Université catholique de Louvain, Belgium. Currently, he is a full professor of computer science at the National Institute of Posts and Telecommunications (INPT), Morocco. His scientific interests include software engineering, conceptual modeling, web semantic, distributed systems, and information systems management.