Reputation Management in Multi-Agent Systems using Permissioned Blockchain Technology

Davide Calvaresi*, Valerio Mattioli†, Alevtina Dubovitskaya*, Aldo F. Dragoni† and Michael Schumacher*
*University of Applied Sciences Western Switzerland, Sierre, Switzerland.
Emails: name.surname@hevs.ch
†Università Politecnica delle Marche, Ancona, Italy.
Emails: name.surname@univpm.it

***Abstract*—The multi-agent framework is a well-known approach to realize distributed intelligent systems. Multi-agent systems (MAS) are increasingly employed in safety- and information-critical domains (e.g., eHealth, cyber-physical systems, financial services, and energy market). Therefore, these systems need to be equipped with mechanisms to ensure transparency and the trustworthiness of the behaviors of their components. Trust can be achieved by employing reputation-based mechanisms. Nevertheless, the existing methods are still unable to fully guarantee the desired accountability and transparency. Aligned with the recent trends, advocating the distribution of trust to avoid the risks of having a single point of failure of the system, this work extends existing efforts on combining blockchain technologies (BCT) and MAS. To attain a trusted environment, we provide the architecture and implementation of a system that allows the agents to interact with each other and enables tracking how their reputation changes after every interaction. Agents reputations are computed transparently using smart contracts. Immutable distributed ledger stores reputation values, as well as services and their evaluations to ensure trustworthy interactions between the agents. We also developed a graphical interface to test different scenarios of interactions between the agents. Finally, we summarize and discuss the experience gained and explain the strategic choices when binding MAS and BCT.**

***Index Terms*—MAS, Trust, Reputation, Blockchain, Smart Contracts**

I. Introduction

Intelligent systems are increasing the pace of their growth and expansion in numerous application areas, also pervading domains dealing with sensitive data such as data privacy, security, and integrity [1], [2]. Multi-agent systems (MAS) are often employed in the architecture design of the intelligent, distributed, and autonomous systems. Although MAS paradigm offers a broad range of capabilities well appreciated even in sensitive domains (e.g., e-health [3], [4], assisted living [5], tele-rehabilitation [6], manufacturing [7], and eCommerce), ensuring privacy, security, and integrity of the data (thus guaranteeing trustworthy and transparent communications) still represent an open challenge [8].

In their models and dynamics, MAS emulate human behaviors, hence, ensuring accountable and trusted interactions between agents is essential but not straightforward. Many remarkable efforts to develop models and mechanisms to guarantee secure communications and trust in MAS have been provided [9], [10], [11]. Yet, evolving application scenarios and technologies constantly demand viable and sustainable solutions addressing as much as possible emerging trust and security requirements. Regardless of the system's scale, both open and private distributed systems can be composed of collaborative and/or competitive agents. Such entities can *(i)* pursue the maximization of their own objectives and plans (e.g., utility function), *(ii)* join and leave at any time the system/community, *(iii)* organize coalitions, and *(iv)* manipulate and exploit each other to foster selfish interests. According to Ramchurn et al. [12], the interaction among agents can be characterized by *(i)* interaction mechanisms, *(ii) who* can take part in a given interaction, and *(iii) when* a given interaction can take place. To reduce the above mentioned risks, to simplify the decision-making process, and to enable autonomous interactions (in terms of "who" and "when") it is necessary to introduce a factor/knowledge (possible sufficient) to classify any agent's reliability.

Reliability, trust and reputation play a key role in MAS since the beginning (in order to revise conflicting beliefs and choose among opinions received from different information sources [13], [14], [15], [16], see also Falcone et al. (2001) for a more general argumentation). According to Ramchurn et al. [12], trust can be formally defined as *"a belief an agent has that the other party will do what it says it will (being honest and reliable) or reciprocate (being reciprocative for the common good of both), given an opportunity to defect to get higher payoffs"*.

A means to establish trusted interactions is to implement *reputation*-based mechanisms. An agent reputation is usually computed on previous behaviors and it can be used to infer about possible future ones. Therefore, in case an agent needs to demand crucial/relevant information, the reputation can heavily impact on the decision making process. Moreover, there is still the need to guarantee that it is properly managed (e.g., to provide a honest and verifiable way to compute it, to ensure that its historical values are tamper-proof, and to make available an updated reputation value).

Recent studies show that binding MAS and blockchain technologies (BCT) are gathering theoretical contributions from many fields [17]. Yet, a quest for a practical, scalable, and privacy-preserving system implementation continues [18]. BCT is a peer-to-peer distributed ledger that provides a transparent and immutable history of all the transactions occurred a given network [19]. The transactions are digitally signed and broadcasted by the participants; they are grouped into blocks in the chronological order, time-stamped, and the content of the block is hashed. A hash serves as the unique block identifier, which is stored in the subsequent block, and can be used to easily verify if the content of the block was modified. The blockchain is replicated and maintained by every participant. With this decentralized approach there is no need for setting up a single trusted centralized entity for managing the registry. The participants will notice a malicious attempt to tamper the information stored in the registry, hence the immutability of the ledger is guaranteed. Thus, the features of the BCT enable us to implement reliable reputation-based mechanisms in MAS. Equipping MAS with BCT will enhance transparency and trust for the reputation management within MAS, overcoming the limitation of current approaches (e.g., need for a trusted third party, a trusted reputation handler and a reliable reputation-computation methods). In particular, even in the case of an unknown nature of the agents, malicious behavior can be timely identified and possibly avoided.

Contributions: This work extends existing efforts on combining BCT and MAS. The novelty relies on structures and mechanisms to compute the agents' reputation. In particular, chaincode functionalities (using Hyperledger Fabric v1.0) have been coupled with the behaviors of a community of agents (realized in JADE - Java Agent DEvelopment Framework). Moreover, the system offers a GUI for managing the agents at both, community and single-agent levels. Both autonomous and user-driven behaviors have been implemented to test the computing mechanisms (smart contracts) using the reputation (agent's heuristics).

The paper is organized as follows: Section II presents and elaborates on the background and related work, Section III provides an overview of the proposed system and describes its architecture, Section IV details the mechanism employed for transparent reputation management, Section V discusses reputation trends and their interpretations. Finally, Section VI concludes the paper.

II. Background and Related Work

This section introduces notions of trust and reputation and presents some works aiming at merging MAS and BCT. Within a community characterized by agents with various and heterogeneous (possibly overlapping) policies, capabilities, and roles, the interactions might be subject to factors such as cost functions, properties, and values evolving over the time. A reasonable assumption is to have agents with a bounded rationale. Therefore, aiming at maximizing their expected utility, agents can only rely on the information (possibly partial) they have [20].

The trust factor can minimize the effects of the uncertainty on the decision-making process. Hence, trust is defined as "a belief an agent has that the other party will do what it says it will (being honest and reliable) or reciprocate (being reciprocative for the common good of both), given an opportunity to defect to get higher payoffs (adapted from (Dasgupta, 1998))" [12]. A way to have a trusted environment is to associate a reputation value per agent, thus being able to discriminate whether it should be part of that community (showing acceptable behaviors) or it should be purged to maintain the community trustworthy.

Reputation can be defined as "a collection of opinions received from other agents" [21]. Generally, it is used to frame the perception and expectation about someone's behavior based on previous interactions (presumably similar to future ones). Such an indicator is an incentive for positive behaviors and it is considered a reliable trust building-block.

Agents' reputation is a pivotal element in trust-based MAS. Depending on the technologies underlying the MAS, several techniques have been employed to ensure trust. Trust can be computed directly by the agents or delegated to a specialized (possibly third-party) entity. Such an attribute can profoundly affect the dynamics of an entire community of agents [21]. Moreover, some corner cases need to be addressed too. According to Racmchurn et al. [12], an agent could provide low-quality services, leave the community as it gets "paid", and then, possibly re-join the system avoiding to be connected to its previous behaviors. A trusted system must prevent such practices. Authenticity, integrity, transparency, and correctness of the reputation, are crucial features to make it effective and reliable. The traditional approaches to manage the reputation are *security-based, institutional*, and *social*, and they operate as follows:

- **security-based:** it usually ensures only integrity and authenticity of data by cryptographic means;
- **institutional:** it assumes the presence of a central authority observing, controlling, and enforcing agents' actions. The vulnerability of such an approach are (i) to be prone to introduce a single point of failure, and (ii) if compromised, the reputation can be tampered and manipulated;
- **social:** it requires an agent able to model other agent behaviors (similarly with the human society). Unfortunately, it is not always applicable.

Finally, even combining such approaches, the above-mentioned requirements cannot be met [17].

BCT can be the key to enhance MAS security, and some attempts at merging these two technologies have already been done [17]. Ferrer [22] discusses the potential advantages and limitations of the combination of blockchain technology and MAS, taking an example of swarm robotic systems. Bottone et al. [23] presented a mathematical structure for a block-less, fee-less, distributed ledger technology employable in wireless sensors networks, Internet of Things, and cyber-physical systems.

BCT provides immutability of its data repositories (ledgers) and the data posted on them [24], and facilitates the process of auditability [25]. Both in research and commercial applications, reputation systems are gaining a stable share [26]. However, such a rapid growth resulted in generating diverging approaches implementing a broad set of solutions. A few solutions associated the reputation and trust in the framework of MAS and BCT have been recently proposed, following the recent trends. For example, Khaqqi et al. [27] proposed an emission trading scheme with a double aim: to reduce emission production and stimulate adoption of long-term abatement technology. Qayumi et al. [28] proposed the introduction of BCT in agentified computing systems dealing with potentially "very-large datasets". Besides several attempts of binding MAS and BCT can be enumerated [17], [29], no actual implementation of computing the agent reputation via smart contracts can be acknowledged yet. Regardless of the strong recent trend towards employing BCT in different areas, where trust management is essential, the question whether a potentially simpler solution can be built around the trusted party remains. For instance, the authors in [30] argue that often participants need to trust each other despite using a distributed ledger. For now we will focus on scenarios when agents have all the same nature (no different a-priori reputations based on their different role in the agency), so that BCT is actually employed to develop trustworthiness of interactions between agents (only) with respect to their behaviors.

The next section presents an architecture which is an extended version of the recent work presented in [29]. Trying to address the latter, and considering that the reputation is usually associated with the identity of a participant, we focus on *permissioned* and *private* BCT implementation. In particular, in the permissioned blockchain, the identities of the agents/users and rights to participate in the consensus (right to write on the ledger and/or validate the transactions) are managed by a membership service. Moreover, we describe the newly developed structures and mechanism computing the agents' reputation over the BCT.

III. Solution Overview and System Architecture

In this section, we present an overview of the proposed system and describe its architecture. Our solution eliminates the need for *assumed* trustworthy agents computing the reputation with *assumed* uniform and unbiased techniques. The reputation management is delegated to smart contracts, thus fully benefiting from the main BCT properties (e.g., data transparency, immutability, integrity).

It is worth to recall that a multi-agent system is composed of atomic autonomous entities (agents) characterized by an extendable knowledge, driven by self-developed or induced objectives able to interact with each other [31]. Such agents are loosely coupled, interconnected, and organized in networks. Regardless of distribution, dimensions, and scope, agent communities might have to take into consideration the possible generation and evolution of undesired/uncontrolled behaviors. Thus, assuming that the agents can possibly manifest malicious behaviors, it has been decided to use the reputation as a discriminating factor to operate in the community. In turn, set a given threshold, the reputation can become a reason for expelling an agent from the community.

JADE has been chosen as the framework employed to develop the MAS given its compliance with the FIPA[3] standard and its simplicity in creating and handling agents [32]. Fabric Hyperledger (v1.0) [33] has been chosen as a permissioned BCT component due to its maturity (development and documentation) and its open-source nature.

A. System Architecture, Agent Identity, and Certificates Management

In the dynamics of the agent community, the two technologies mentioned above get firstly connected in the *management of the identities of the agents*. To be able to interact with each other and operate on the ledger, the agents need to register and obtain the credentials, including certificates of the corresponding public keys. Therefore, two main classes of agents have been created:

BC-A : regular agent operating in a given community, in which all the interactions are recorded on the blockchain;

CA-A : agent handling the registration in a given agents community. In particular, the CA-A is in charge of interacting with the certification authority (CA) component of Hyperledger; the CA-A also offers a possibility of encoding rules and conditions for the enrollment. In the settings of multiple CAs, available in more recent versions of Hyperledger, similar multi-signature approaches can be employed to manage mutliple CA-As.

Although having different functions, both CA-A and BC-A have the same structure (Figure 1). The agent is composed of:

- core: the agent instance in JADE,
- view: the functionality and user interface details,
- behavior: the list of possible actions,
- model: the adapter to operate on the underlying BCT,
- controller: the component connecting view and model.

The BCT components are:

- Certification Authority (CA): an entity providing valid identities and certificates for the members of the blockchain network.
- Membership service: an entity that identifies CA(s) trusted to define the members of the network. An MS can identify specific roles an actor might play either within the scope of the network and sets the basis for defining access privileges [34].
- Ledger: the immutable, sequenced, tamper-resistant *chain of blocks* that records all the transactions and the database state.
- Chaincode: a program (also known as smart contract) that is developed to interact with the ledger.
- Invoke: the APIs called when the invoke transaction is received, in this case from the agents, to process any transaction proposals of assets.
- Assets: a collection of key-value pairs recorded as transactions in the ledger and the respective functions.

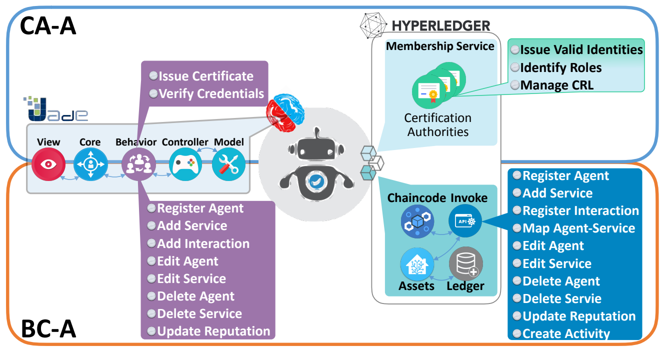


Fig. 1. Conceptual design of the system components.

In this implementation, every agent ($BC - A_i$) hosts a peer that maintains the ledger and executes the chaincode. However, agents and peers can also be decoupled over the network.

B. The Ledger: Agents, Reputations, Services and Interactions

Conceptually, the ledger stores *(i)* the information about the service(s) provided by the agents[1], *(ii)* the information about the interactions that took place in the community, and *(iii)* the related evaluation from both service's demander and executor.

It is worth to recall that in Hyperledger Fabric v1.0 the ledger consists of two distinct, though related, components: *(i)* World State database (to maintain the current state of a set of ledger states) and *(ii)* Blockchain - immutable sequence of blocks, each of which contains a set of ordered transactions. Concerning the World State database, two alternatives technologies were available: CouchDB and LevelDB (used in the presented solution). CouchDB allows "richQueries" but cannot prevent "phantom reads". Although LevelDB does not offer the possibility to directly perform "richQueries", it has a relational data model, support SQL queries, and provide support for the indexing, it is not affected by "phantom reads" and can overcome the mentioned limitation by using

[1]in the form of a tuple: {*agent; service(s); additional info*}

composite keys as indices. Thus, the relational mapping has been handcrafted. Figure 2 shows the representation of the system's elements in the DB.

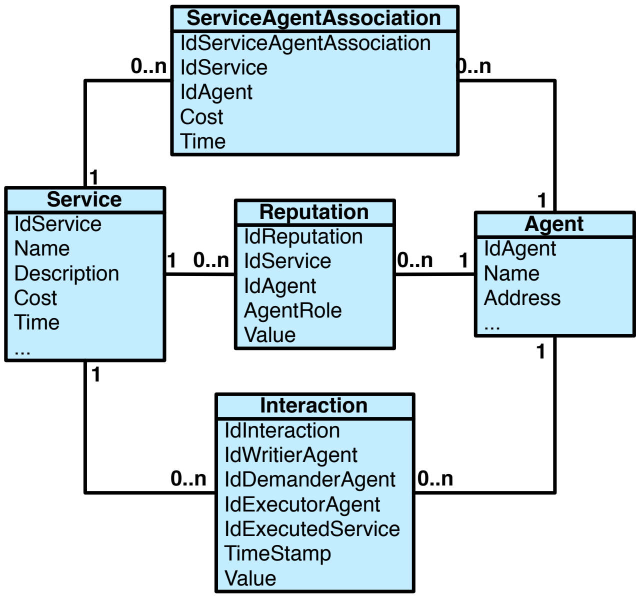


Fig. 2. Relational schema of the implementation on levelDB.

LevelDB saves the tables in `json` format. Listing 1 shows the implementation in GO[2] of the reputation table.

```
type Reputation struct {
 ReputationId    string `json:"ReputationId"`
 AgentId         string `json:"AgentId"`
 ServiceId       string `json:"ServiceId"`
 AgentRole       string `json:"AgentRole"`
 Value           string `json:"Value"`
}
```

Listing 1: Reputation Structure in the Ledger.

To allow the Level-DB World State to execute complex queries, we created a composite keys-indices mechanism. In particular, if an agent is offering a given service for the first time, the mechanism is triggered during its publication (assigning a "neutral" starting value: 6/10). If an agent is demanding a given service for the first time, such a mechanism occurs after the creation of the record `interaction`.

In the developed system, the composite key is characterized by `agentId – serviceId – agentRole – reputationId`; this key is the pointer to a given value (address: reputationId) in the DB. This enables execution of the queries such as *"SELECT Agent(s) WHERE (...)"* (partial-key queries). Moreover, if more complex queries are necessary, the system executes partial-keys and simple queries in cascade. Such an index is not required to be associated with a value. Hence, the queried values can be extracted from the composite-index itself.

[2]https://golang.org/

When creating the composite keys-indices, the agent role (demander/executor) is checked first. Then, the previous (or initial) value of its reputation is accessed. Finally, given the evaluation of an interaction, the agent reputation is updated accordingly (see Listing 2).

```
func CreateAgentServiceRoleIndex(reputation *Reputation, stub
shim.ChaincodeStubInterface) (agentServiceRoleIndex string, err error){
        indexName := "agent˜service˜agentRole˜reputation"
        agentServiceRoleIndex, err = stub.CreateCompositeKey(indexName,
        []string{reputation.AgentId, reputation.ServiceId,
        reputation.AgentRole, reputation.ReputationId})
        if err != nil {return agentServiceRoleIndex, err}
    return agentServiceRoleIndex, nil}
```

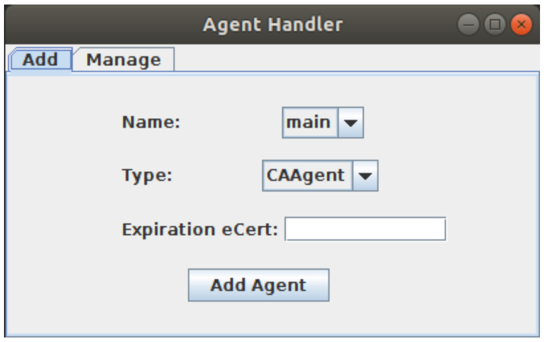
Listing 2: Reputation composite keys-indices creation.

In JADE, the entity connecting *an agent* and *a service(s) offered by the agent* is called Directory Facilitator (DF)[3]. This entity and the respective functions have been replaced by the BCT. This design choice enables to:

- avoid a single point of failure (if the DF is unique in the community),
- reduce the response time when inquired by regular agents,
- improve accessibility and transparency,
- ensure immutability and traceability.

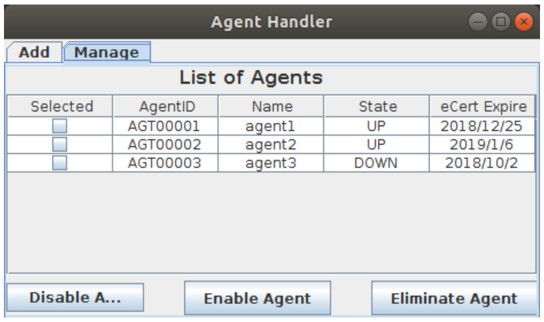
C. GUI

The agents' behaviors are both automated and manual (action and choices are delegated to the user). To manually interact with the agents, and therefore with the blockchain, there is the classic command line interface and a customizable graphical interface. Figure 3 shows the interface that enables to create and register the agents (needed to test the platform). Figure 4 presents the interface to manage such agents (e.g., enabling, disabling, and eliminating agents).



Name:	main
Type:	CAAgent
Expiration eCert:	

Fig. 3. View to *add* agents into the community.



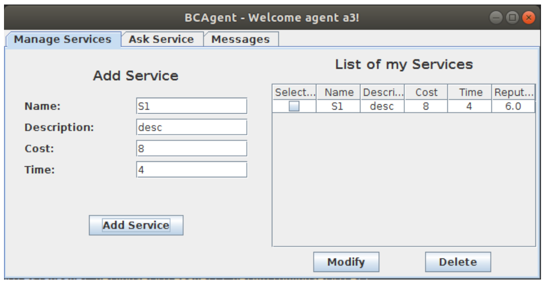
List of Agents

Selected	AgentID	Name	State	eCert Expire
☐	AGT00001	agent1	UP	2018/12/25
☐	AGT00002	agent2	UP	2019/1/6
☐	AGT00003	agent3	DOWN	2018/10/2

Fig. 4. View to *manage* the agents.

[3] http://www.fipa.org/specs/fipa00023/

Figure 5 shows the interface that enables to add and modify the services that a given agent is offering. To require a service execution, we provide the interface depicted on Figure 6. A pre-filtering and sorting can be applied by selecting the optional features (e.g., cost, time, and reputation). Finally, an interface developed to accept or reject requests, to communicate with other agents, and negotiate the interface to compose and manage the messages is shown on Figure 7.


BCAgent - Welcome agent a3!

Manage Services | Ask Service | Messages

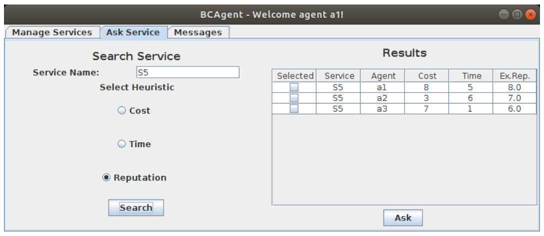
Add Service

Name: S1
Description: desc
Cost: 8
Time: 4

List of my Services

Select...	Name	Descri...	Cost	Time	Reput...
☐	S1	desc	8	4	6.0

Fig. 5. View to *add* services into the agent.


BCAgent - Welcome agent a1!

Manage Services | Ask Service | Messages

Search Service

Service Name: S5

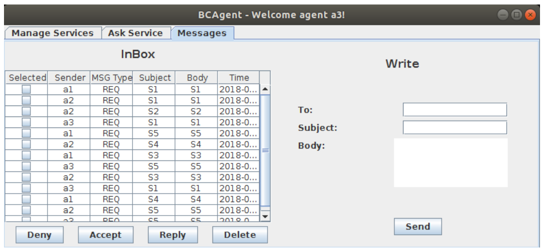
Select Heuristic

Cost [✓ Reputation] Time

Results

Selected	Service	Agent	Cost	Time	Ex.Rep.
☐	S5	a1	8	5	8.0
☐	S5	a2	3	6	7.0
☐	S5	a3	7	1	6.0

Fig. 6. View to *search* and *ask* for a service.


BCAgent - Welcome agent a3!

Manage Services | Ask Service | Messages

InBox

Selected	Sender	MSG Type	Subject	Body	Time
☐	a1	REQ	S1	S1	2018-0...
☐	a2	REQ	S1	S1	2018-0...
☐	a2	REQ	S2	S2	2018-0...
☐	a3	REQ	S1	S1	2018-0...
☐	a1	REQ	S5	S5	2018-0...
☐	a2	REQ	S4	S4	2018-0...
☐	a1	REQ	S3	S3	2018-0...
☐	a3	REQ	S5	S5	2018-0...
☐	a2	REQ	S3	S3	2018-0...
☐	a3	REQ	S1	S1	2018-0...
☐	a1	REQ	S4	S4	2018-0...
☐	a2	REQ	S5	S5	2018-0...
☐	a3	REQ	S5	S5	2018-0...

Write

To:
Subject:
Body:

Fig. 7. View to *reply* and *send* messages.

IV. Interactions and Reputation Management

Features such as tracking the service evolution over the time (e.g., costs, offered performance, and availability) can be easily introduced, thus enabling to employ counter-measures in case of speculations or selfish behaviors. Therefore, we decided to associate the concept of reputation with every agent per service, as an evaluator in the role of demander and/or executor. The implemented architecture enables to provide:

- an *overall reputation value* rating the general (average) agent's reputation;
- a *task specific value* of a given service and role (demander/executor).

Once an agent is registered, it can require and/or provide some services. The reputation is initially set to a default value. Assuming direct interactions (e.g., CNET based) both the agents (demander and executor) must be able to evaluate the output of the interaction at its completion.

Then, once received both evaluations, a smart contract is triggered and the executor is notified with the evaluation provided by the demander. If such an evaluation is significantly different from the one provided by the executor and the latter accepts to revise it accordingly, the new reputation value is computed and the ledger is updated (Figure 8(a)). Otherwise, if the difference between the evaluation overpasses a given threshold, a disagreement resolution process is started (Figure 8(b)).

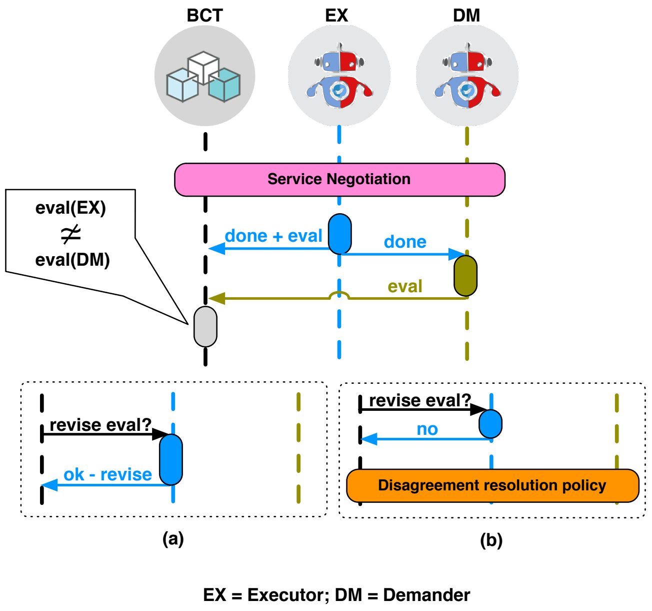


Fig. 8. Conceptual design of the system components.

Disagreements and conflicts can happen due to the autonomy and heterogeneity of the MAS components, as well as due to possible malicious behaviors. A considerable number of scientific contributions cope with *conflict resolution* in the MAS frameworks [35], [36], [37]. Different approaches can be employed based on the particular scenarios of using MAS, the model of expected behavior, and outcomes of individual agents. Currently, in our system, we consider the two following scenarios:

- agreement on a specific value (with a minor variance),
- disagreement (the evaluations of demander and executor have a misalignment greater than a customizable threshold).

In the case of disagreement, we employ the simple approach that consists of the following steps: *(i)* proposing the agents to revise the evaluations (enables to identify unintentional mistakes) and *(ii)* consulting another agent with a higher reputation. One has to notice that it is possible to employ other existing approaches and policies for disagreement resolution.

To handle the disagreement resolution the system provides APIs at the agent level (JADE). By doing so, it is possible to trigger several investigations upon the disagreeing values. From the smart contracts side, the function provided to support the investigations is to check the evolution (over the time) of the reputations of both demander and executor. It enables detection of the behavior patterns that can lead to malicious trends, systematic errors, or just to a single fault.

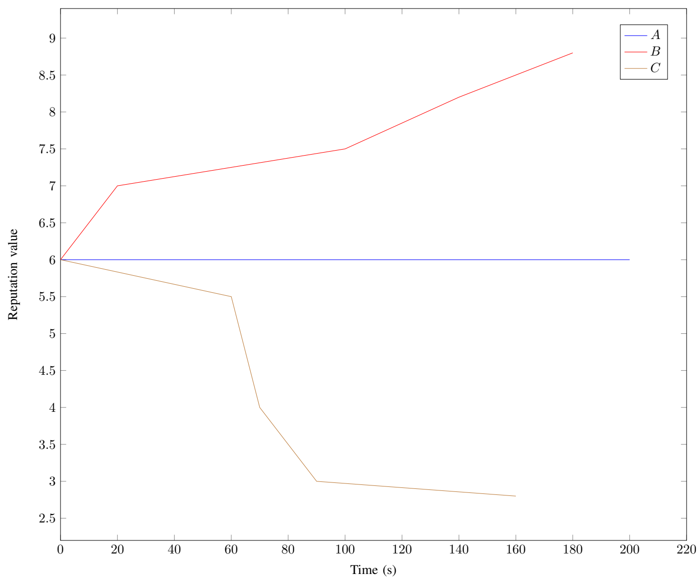


Fig. 9. Identified basic reputation trends.

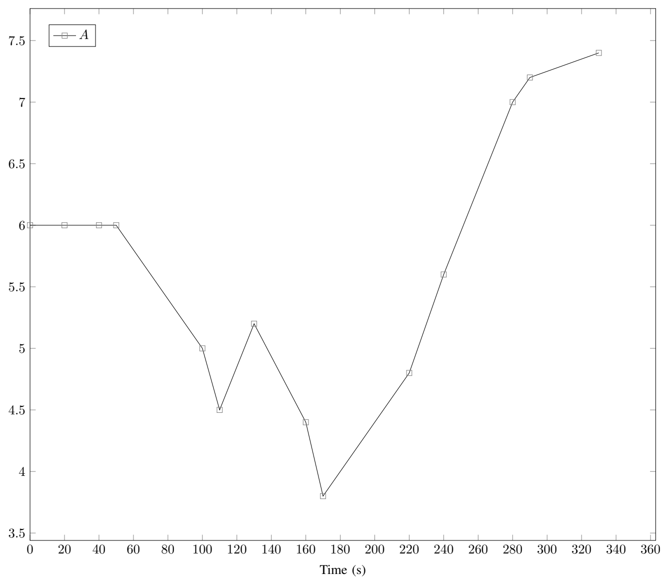


Fig. 10. Identified composite reputation trends.

V. Discussion

In this section, we discuss how we interpreted agents' behavior and reliability using the transparent reputation management implemented in our system. First, we present possible agent's behavior trends, second, we discuss how they can be interpreted based on the agent's role (demander or executor of a service). Finally, we discuss the cases of missing evaluations and their potential impact on the agents' reputations.

Figure 9 shows basic reputation trends that can be used to characterize the agent's behavior. We assume that at the moment of the registration every agent gets a "fair" reputation value of 6 out of 10. Implementing several dynamics and varying the nature of the agents' behaviors, we have been able to identify three simple types of trends (see Figure 9).

Figure 9 - A corresponds to the case when the reputation remains constant. Such behavior can occur if *(i)* the agent's performance is stable, or *(ii)* there was no activity (i.e., no service requested or provided, or the agent was off-line). Figure 9 - B depicts the case when the reputation values are increasing and approaching the maximum reputation value threshold (10 in our simplified example). In Figure 9 - C the reputation of the agent decreases, approaching the lower bound threshold. The latter, combined with the inclination of the curve, can be used when taking the decision of excluding the agent from the community.

Figure 9 shows the behavioral trends, however, the actual reputation may be a composition of the basic trends presented above. Figure 10 shows a possible scenario, where each point of the graph corresponds to the punctual reputation value based on the evaluations provided by both demander and executor concerning a given interaction. Evaluation methods and quality of the performance offered by the agents can present considerable variability which is reflected in the fluctuations of the agents' reputations.

The behavioral trends can be interpreted based on the agent role. Table I summarizes the possible interpretations for the executor and demander for every trend identified and discussed above. The trends are significantly easier to interpret. For instance, if the reputation value of the executor grows over the time, this agent can be seen as a reliable service provider, constantly improving the quality of its service. However, in the case of the composition of the trends, interpretation may not be straightforward. Yet, outliers perturbing the reputation's trend can be used to detect malicious behavior, a single mistake of the service provider, or unintentionally made errors when providing evaluations. The relevance of the trend can be interpreted based on the time (for how long a given value is maintained) and on the number of evaluation (number of interactions and evaluations occurred). The combination of the slope of a curve with the number of points also plays an important role. It can be used to detect unintentional mistakes in evaluation, as well as inactivity of an agent.

	Fig. 9(A)	Fig. 9(B)	Fig. 9(C)	Fig. 10(A)
Exe	Stable	Bad evaluators(s), Performance in decline	Good Executor	Unpredictatble, Time-limited problem
Dem	Stable	Bad executor(s), Possible malicious intentions	Good evaluator	Honest mistake, Single bad action

TABLE I
Possible interpretations of the identified reputation trends.

Currently, we are analyzing potential corner cases that can occur when our system is in use. These include a missing evaluation from a demander (currently, there is no mechanism to enforce the demander to provide an evaluation), an executor (it can be off-line, and therefore the evaluation will never be provided), or both. We are also investigating a possibility to put timing constraints for providing evaluations, in particular for the executor. Introduced in a smart contract, such constraints will enable identification of the situations when the deadline of the execution of the service is missed. Thus, the reputation has to be updated accordingly.

VI. Conclusions and Future Work

This paper addressed the challenge of enforcing trust in multi-agent systems by implementing mechanisms for transparent reputation management using block- chain technology. We present the design and development of a system based on JADE and Fabric Hyperledger v1.0.

Our system achieves a trustworthy community by extending the management of the agent identity to the membership service of Hyperledger Fabric, by distributing the association agent - offered service(s) using the ledger, and by implementing the mechanisms to immutably store and transparently compute agents' reputations based on the evaluations of the interactions between the agents. Several behaviors of the agents were simulated to test the proposed system. The results show promising directions to undertake. The system appears robust and scalable. Moreover, its graphical interface simplifies the interactions making the tests much faster and easier with respect to a classic command line interface and providing a possibility to employ this system in various use-case scenarios. Extending the interface will enable us to obtain extensive evaluation results for specific use-case scenarios and interactions.

The application of relatively new technology such as BCT are challenging. This latter is not fully framed by standards and has not been widely adopted yet. Although the benefits of combining BCT and MAS are justified and a prototype of the proposed solution has been implemented, some technical limitations persist. For example, a few challenges that still need to be addressed are: *(i)* avoiding single point(s) of failure of the system by defining a reasonable mapping between real entities and distributed components (membership service, orderer) of the underlying blockchain technology (Hyperledger Fabric), *(ii)* leveraging cryptographic solutions for providing better security and privacy, *(iii)* verifying the correctness of the implementation of the smart contract, and *(iv)* adopting and adapting the blockchain and agent technology in the real-world systems.

References

[1] H. C. Wong and K. Sycara, "Adding security and trust to multiagent systems," *Applied Artificial Intelligence*, vol. 14, pp. 927–941, 2000.

[2] W. He, X. Gao, W. Zhong, and F. Qian, "Secure impulsive synchronization control of multi-agent systems under deception attacks," *Information Sciences*, 2018.

[3] D. Calvaresi, A. Claudi, A. Dragoni, E. Yu, D. Accattoli, and P. Sernani, "A goal-oriented requirements engineering approach for the ambient assisted living domain," in *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA '14, 2014, pp. 20:1–20:4. [Online]. Available: http://doi.acm.org/10.1145/2674396.2674416

[4] A. Dubovitskaya, V. Urovi, I. Barba, K. Aberer, and M. I. Schumacher, "A multiagent system for dynamic data aggregation in medical research," *BioMed research international*, vol. 2016, 2016.

[5] D. Calvaresi, D. Cesarini, P. Sernani, M. Marinoni, A. Dragoni, and A. Sturm, "Exploring the ambient assisted living domain: a systematic review," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–19, 2016.

[6] D. Calvaresi, M. Schumacher, M. Marinoni, R. Hilfiker, A. Dragoni, and G. Buttazzo, "Agent-based systems for telerehabilitation: strengths, limitations and future challenges," in *proceedings of X Workshop on Agents Applied in Health Care*, 2017.

[7] F.-S. Hsieh, "Modeling and control of holonic manufacturing systems based on extended contract net protocol," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 6, 2002, pp. 5037–5042.

[8] Y. Hedin and E. Moradian, "Security in multi-agent systems," *Procedia Computer Science*, vol. 60, pp. 1604–1612, 2015.

[9] B. Yu and M. P. Singh, "An evidential model of distributed reputation management," in *Proceedings of 1st international conference on Autonomous Agents and Multiagent Systems*. ACM, 2002, pp. 294–301.

[10] S. Ramchurn, D. Huynh, and N. Jennings, "Trust in multi-agent systems," *The Knowledge Engineering Review*, p. 1–25, 2004.

[11] Y. Hedin and E. Moradian, "Security in multi-agent systems," *Procedia Computer Science*, vol. 60, pp. 1604 – 1612, 2015, knowledge-Based and Intelligent Information and Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings.

[12] S. D. Ramchurn, D. Huynh, and N. R. Jennings, "Trust in multi-agent systems," *The Knowledge Engineering Review*, vol. 19, pp. 1–25, 2004.

[13] A. Dragoni, F. Mascaretti, and P. Puliti, "A generalized approach to consistency based belief revision," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 992, pp. 231–236, 1995.

[14] A. Dragoni and P. Giorgini, "Distributed belief revision," *Autonomous Agents and Multi-Agent Systems*, vol. 6, no. 2, pp. 115–143, 2003.

[15] A. Dragoni and S. Animali, "Maximal consistency, theory of evidence, and bayesian conditioning in the investigative domain," *Cybernetics and Systems*, vol. 34, no. 6-7, pp. 419–465, 2003.

[16] A. Dragoni and P. Giorgini, "Learning agents' reliability through bayesian conditioning: A simulation experiment," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1221, pp. 151–167, 1997.

[17] D. Calvaresi, A. Dubovitskaya, J. P. Calbimonte, K. Taveter, and M. Schumacher, "Multi-agent systems and blockchain: Results from a systematic literature review," 2018.

[18] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *International Workshop on Open Problems in Network Security*. Springer, 2015, pp. 112–125.

[19] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[20] L. J. Savage, *The foundations of statistics*. Courier Corporation, 1972.

[21] F.-S. Hsieh, "Modeling and control of holonic manufacturing systems based on extended contract net protocol," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 6, 2002, pp. 5037–5042.

[22] E. C. Ferrer, "The blockchain: a new framework for robotic swarm systems," *arXiv preprint arXiv:1608.00695*, 2016.

[23] M. Bottone, F. Raimondi, and G. Primiero, "Multi-agent based simulations of block-free distributed ledgers," 2018.

[24] A. Zuiderwijk, M. Janssen, and C. Davis, "Innovation with open data: Essential elements of open data ecosystems," *Information Polity*, vol. 19, no. 1, 2, pp. 17–33, 2014.

[25] S. Kozlowski, "An audit ecosystem to support blockchain-based accounting and assurance," in *Continuous Auditing: Theory and Application*. Emerald Publishing Limited, 2018, pp. 299–313.

[26] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision support systems*, vol. 43, no. 2, pp. 618–644, 2007.

[27] K. N. Khaqqi, J. J. Sikorski, K. Hadinoto, and M. Kraft, "Incorporating seller/buyer reputation-based system in blockchain-enabled emission trading application," *Applied Energy*, vol. 209, pp. 8–19, 2018.

[28] K. Qayumi, "Multi-agent based intelligence generation from very large datasets," in *Cloud Engineering (IC2E), 2015 IEEE International Conference on*. IEEE, 2015, pp. 502–504.

[29] D. Calvaresi, A. Dubovitskaya, D. Retaggi, A. Dragoni, and M. Schumacher, "Trusted registration, negotiation, and service evaluation in multi-agent systems throughout the blockchain technology," *International Conference on Web Intelligence*, 2018.

[30] T. Locher, S. Obermeier, and Y.-A. Pignolet, "When can a distributed ledger replace a trusted third party?" *preprint arXiv:1806.10929*, 2018.

[31] S. J. Russell and P. Norving, "Norvig," *Artificial Intelligence: A Modern Approach*, pp. 111–114, 2003.

[32] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*. John Wiley & Sons, 2007, vol. 7.

[33] C. Cachin, "Architecture of the hyperledger blockchain fabric," in *Proc. of Distributed Cryptocurrencies and Consensus Ledgers*, 2016.

[34] IBM, "hyperledger-fabric online documentation," https://hyperledger-fabric.readthedocs.io/en/release-1.2/, [Accessed 10/9/18].

[35] C. Tessier, L. Chaudron, and H.-J. Müller, *Conflicting agents: conflict management in multi-agent systems*. Springer Science & Business Media, 2006, vol. 1.

[36] S. Resmerita and M. Heymann, "Conflict resolution in multi-agent systems," in *IEEE Conference on Decision and Control*, vol. 3, 2003, pp. 2537–2542.

[37] W. W. Vasconcelos, M. J. Kollingbaum, and T. J. Norman, "Normative conflict resolution in multi-agent systems," *Autonomous agents and multi-agent systems*, vol. 19, no. 2, pp. 124–152, 2009.