Incorporating Trust into Collaborative Social Computing Applications

Paula Muñoz[1], Alejandro Pérez-Vereda[2], Nathalie Moreno[1], Javier Troya[1], and Antonio Vallecillo[1]

[1]*ITIS Software. Universidad de Málaga, Spain*
[2]*Universidad de Castilla-La Mancha, Spain*
{paulam,nmv,jtroya,av}@uma.es, alejandro.pvereda@uclm.es

***Abstract*—Mobile-based collaborative social computing applications represent an alternative to the server-centric models currently offered by major IT vendors, where users own the information they generate and take control over how other users access and exploit it. In this context, trust is fundamental for sharing information and making decisions based on the data collected from other users. This work develops a trust management system embedded in the Digital Avatars framework for collaborative social computing applications, using Subjective logic. It enables explicit representation and operation with trust information about both service providers (functional trust) and other users' opinions about these providers (referral trust) in peer-to-peer environments. The proposal is specified using high-level models that can also be simulated and validated, and serve as a guide for the corresponding implementations in an existing social computing application platform. The proposed solution has been successfully applied in a collaborative carpooling system, where users need to trust other users with whom they share travels.**

***Index Terms*—collaborative social computing applications, uncertainty, trust.**

I. Introduction

Social computing (SC) is the area of Information Technology (IT) that deals with the interrelationships between social behavior and computer systems. Originally focused on the processing and analysis of social information, the term has progressively evolved towards a wider meaning that includes the use of computer systems to support any type of social behavior, and where humans become the main protagonists, not only as beneficiaries but also as active players.

To support social computing, current systems are mostly based on server-centric applications, where content created by distributed users is transferred to cloud servers. This is, for example, the model commonly used in most social applications from Google, Facebook, Amazon and the rest of the big IT players. This server-centric approach is also widely exploited in smart city applications, a domain where social computing is rapidly gaining relevance.

One of the problems of this centralized architectural model is that the users' personal information is owned and managed by the applications' vendors, and users lose control over it. An alternative approach advocates the adoption of collaborative peer-to-peer models based on mobile devices, e.g., smartphones or tablets, as the main components of the system architecture. This is also known as *mobile-based collaborative social computing* (MCSC) applications. This model of collaborative computing enables the empowerment of users, allowing them to take control of the information and contents they generate, and how all that information is accessed and exploited in a secure manner by third parties. Our proposal leverages the capabilities currently offered by mobile phones, which are widely-used devices, using the Digital Avatars collaborative framework. This framework is based on the People-as-a-Service (PeaaS) model [1], which promotes the user to become a service provider with her own information. This model has been presented in previous works and already used in contexts such as the Internet of Things (IoT) [2], smart cities [3] or gerontology [4].

With the Digital Avatars collaborative framework, we introduce the concept of a Digital Avatar (DA), an entity residing in a person's smartphone or tablet that records information about the owner and offers different services for the interaction with the environment and with the DAs of other users, ensuring the levels of privacy and security dictated by their owners — i.e., DAs serve as smart proxies for them. Because users of such mobile social network applications may not have had any previous interactions, it is important to establish an acceptable level of trust relationships among them. This is critical for sharing information and for making decisions based on the data collected from other users. In this context, trust becomes an aspect of key relevance.

Trust management in peer-to-peer social networks has not received much attention yet. Its treatment is much more difficult than in traditional centralized environments, mainly due to the absence of a central authority, the dynamic topology of the network, and the lack of a global view of the system. These factors limit the trust management process to local information, resulting in uncertainty [5] and incompleteness of trust. In addition, there may be malicious users who provide fake or erroneous information, so trust establishment would be based on incomplete and incorrect data [6]. Other restrictions of these applications are due to the limitations of system resources and network connectivity, which require the use of lightweight algorithms and techniques.

In this paper, we develop a trust management system for Digital Avatars in the context of collaborative social computing applications. We show how the records of a DA can be enriched with confidence about their truthfulness (the so-called *functional trust*) and the information coming from other users'

DAs can be further qualified with the degree of confidence we have about these users' opinions (*referral trust*). We use Subjective Logic [7] to represent both types of trust, for two main reasons. First, subjective logic extends probabilistic logic with information concerning the level of ignorance we have about a statement, providing richer reasoning mechanisms to arrive at informed decisions, as they consider not only the degree of belief and disbelief, but also the degree of uncertainty. Second, subjective logic provides some useful operators to deal with both functional and referral trusts, allowing their smooth and effective combination.

We have developed a prototype of the Digital Avatars framework with the trust management system integrated to serve as a proof-of-concept for the proposal and to evaluate its advantages and limitations. An exemplary application of a collaborative carpooling system has been developed using the framework, and the results are discussed in this paper.

The structure of this document is as follows. After this introduction, Section II briefly describes the background of our work and presents the example that is used to motivate our proposal. Then, Section III describes our work and the implementation we have developed of the framework. Finally, Section IV relates our work to other similar approaches and Section V concludes and with an outline of future work.

II. Background

To set the paper terminology, this section briefly describes the context of the work and the main concepts used in the paper. We also introduce a running example that illustrates how DAs are used in collaborative applications.

A. Digital Avatars

The Digital Avatars framework is a realization and extension of the PeaaS model [1]. PeaaS provides a conceptual framework for application development focused on the smartphone as a representative and interface to its owner. A DA contains a set of *records* that store the persistent information defined by the user, captured by the smartphone sensors and external devices, and by the different applications the user participates in. All this information is stored locally in the smartphone, ensuring that its owner keeps full control over which data is being shared and with whom. This information is the target of the interaction services provided by the Digital Avatars framework. Using the framework, third parties are able to generate value-added interfaces for their own services, resulting on dynamic interactions that are always under the control of the data owner

The information is partly acquired by the Digital Avatars' Complex Event Processing (CEP) engine [8] able to dynamically execute rules capable of processing events coming from different sources and perform actions accordingly. Sources include raw data collected by the smartphone's sensors or by other devices connected to it, as well as events produced by the CEP rules themselves, by applications installed in the smartphone, or by the CEP engines of other external users' DAs. One of the main advantages of CEP is that it works in real time, reducing latency in the decision-making process.

Digital Avatars applications are then specified and developed in terms of a set of CEP rules that are installed in the smartphone engine and handle the received events. Applications implement their behavior by executing the CEP rules and performing read and write operations on the Digital Avatar, which implements an API to manage the DA records. The Digital Avatar API is the second service (apart from the CEP engine) offered by the DA framework to application developers for interacting with the DA.

B. Trust

In our context, trust can refer to the degree of confidence we have either in people or in things. The first one refers to the degree of reliability (trustworthiness) we assign to people to perform an action (*functional* trust) or to report about the reliability of other people (*referral* trust) [7], [9]. Trust in things refers to the level of certainty we assign to them, for example, the accuracy of information stored in a data record, the precision of a measurement, or the degree of belief about the occurrence of an event. We shall call *confidence* the trust we have in things [10].

When specifying trust in people, we need to identify two parties. The *Truster* is the party that states its degree of trust in the *Trustee*, a second entity who is supposed to provide the required service [11]. Such a relationship does not necessarily have to be one-to-one, but could also be one-to-many, and does not need to be either mutual or symmetric [12].

Trust is also context-dependent, which means that a trustee does not need to be trusted in all situations. For example, Ada may trust Bob as a reliable driver, but she does not trust Bob's ability to look after her pets. Therefore, trust is not absolute, but must be specified within a *scope* [7], [12].

Finally, trust in people is *subjective*, i.e., it depends on the truster, and is normally conditioned by uncertain factors [13]. As stated in [14]: "Trust is a psychological state involving positive confident expectations about the competence, benevolence, integrity and predictability of another person and willingness to act on the basis of these expectations. Issues of trust arise in contexts that involve risk, vulnerability, uncertainty and interdependence. Trust expectations are created primarily by the interaction of the perceived qualities of the trustee and contextual factors in play when trust decisions are made."

C. Confidence

As mentioned earlier, we need to distinguish between the trust in people and the trust we place in things, that we shall call *confidence* [10], [15], [16], and which is caused by uncertainty. Here, by uncertainty we mean "the quality or state that involves imperfect and/or unknown information. It applies to predictions of future events, estimates, physical measurements, or unknown properties of a system" [17].

From a generic decision-making perspective, confidence is the degree of belief in a given hypothesis [15] and this is why we will use the term *confidence* to refer to the degree of belief

that a person (the truster) has in something. For example, the confidence that Bob assigns to the readings of the temperature or humidity sensors of his room. In our context, confidence can be used by a DA to assign degrees of beliefs about the truthfulness of the information stored in its records, if required. The treatment of confidence in software models was already described in some of our previous works [10], [18], [19], and therefore we will not consider it further in this paper.

D. Subjective logic

Traditionally, degrees of trust or confidence have been modeled using numbers between 0 and 1 that represent probabilities, and reasoning about trust has been accomplished using probability theory [20], [21]. However, this approach has some limitations, especially when it comes to representing subjective opinions for which users cannot easily express their uncertainty, e.g., their ignorance about the facts they are considering, or their inability to assign an accurate probability to a fact. For example, when the user has total ignorance about some statement x, it might be preferable to say "I don't know" than assigning x a confidence of 0.5. In general, forcing users to set probabilities to express their opinions could lead to unreliable conclusions [19]. This is when Subjective logic comes into play.

Subjective logic, by Audun Jøsang [7], is an extension of probabilistic logic that explicitly takes uncertainty into account. Subjective opinions express beliefs about the truth of propositions under degrees of uncertainty. They can also indicate confidence, or trust, on a given statement and this is what makes them suitable in our context.

Let x be a Boolean predicate. A binomial *opinion* about the truth of x is defined as a quadruple $\omega_x = (b_x, d_x, u_x, a_x)$ where:

- b_x (*belief*) is the degree of belief that x is true.
- d_x (*disbelief*) is the degree of belief that x is false.
- u_x (*uncertainty*) is the degree of uncertainty about x, i.e., the amount of uncommitted belief.
- a_x (*base rate*) is the prior probability of x.

These values satisfy the constraints that $b_x + d_x + u_x = 1$, and $b_x, d_x, u_x, a_x \in [0, 1]$.

Intuitively, the *base rate* of an opinion represents the *objective* probability that can be assigned to the statement using *a priori* evidences or statistical estimates, whilst the other elements of the tuple represent the *subjective* degrees of belief, disbelief and uncertainty about the statement assigned by the expert. Thus, regardless of the value of the prior probability, different belief agents can express their subjective opinions about the statement, including their degree of uncertainty. This is precisely what allows different experts to simultaneously express their individual opinions on the same fact, something common in most collaborative systems where separate users interoperate with each other to achieve their goals.

In addition to the traditional logical operators (and, or, implies, etc.) used to combine the opinions of the same expert about different truth statements, Subjective logic implements *fusion* operators for combining the subjective opinions of different users about the same statement. The goal is to produce a single opinion that better reflects the collection of opinions, or is closer to the truth than each opinion in isolation. This is essential for permitting collaborative modeling and enabling cooperative work between users when they need to reach agreements about how to proceed.

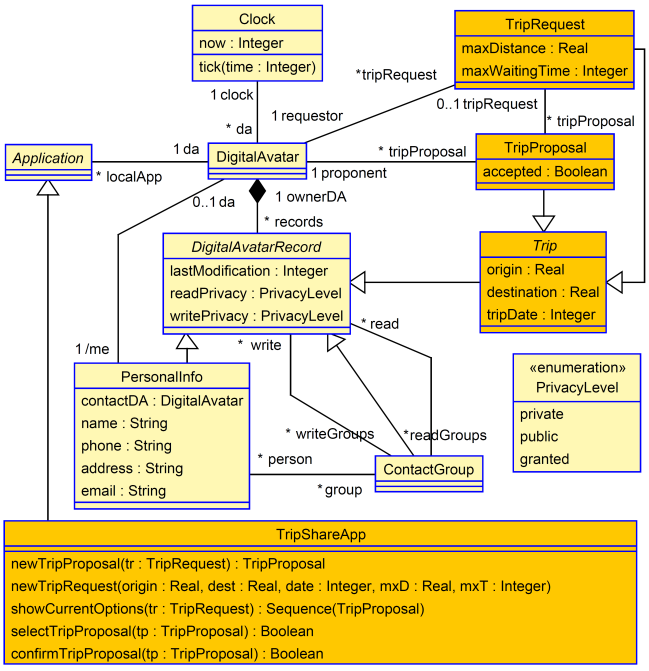

Figure 1: Class Diagram of the Carpooling application.

To represent and operate with Subjective logic values, in [19] we defined the new primitive datatype SBoolean that extends type Boolean with uncertainty information, and provides all corresponding operations. A SBoolean value is defined by the quadruple (b, d, u, a) that represents the corresponding opinion in Subjective logic. The embedding of a Probability c representing a confidence into type SBoolean is achieved by assigning the opinion $w_x = (c, 1 - c, 0, c)$ to x. Considering the embedding of type Boolean into Probabilities, we have that Boolean values true and false correspond, respectively, to opinions $(1, 0, 0, 1)$ and $(0, 1, 0, 0)$. Examples of the use and application of Subjective logic in models represented with UML/OCL can be found in [19].

E. A motivating example

The UML model of our running example of a carpooling system is displayed in Fig. 1. Let us focus for now on the shaded classes — the remaining classes model the Digital Avatars framework, as explained in Section III-A.

Suppose that Bob is a university student who uses the bus every day to get to the university and back to his hometown. Today, Bob has finished his classes earlier than usual and the next bus does not leave for another two hours. Therefore, he decides to use his DA to launch a query to find alternative means of transportation. From the responses received from the DAs predicting a similar trip by car this afternoon, Bob

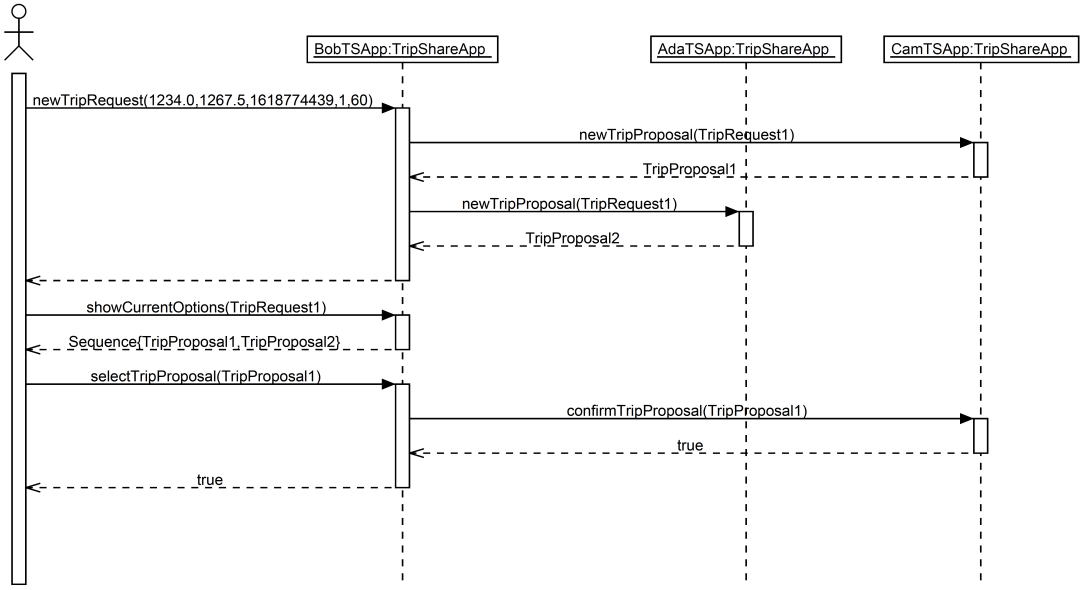

Figure 2: An exemplar Sequence Diagram of the carpooling application.

selects the one from Cam, since she is expected to leave the campus soon and they both live nearby. Next, their DAs ask each other if they are willing to make the trip together.

This app is modeled by class TripShareApp (cf. Section III-A). It manages three types of records: a Trip defines the trip origin, destination, and departure date and time; a TripRequest also specifies the requesting DA (requestor) as well as the maximum distance and time the requestor is happy to accept as deviations from the original request; finally, a TripProposal specifies the details of the proponent's trip, and a reference to the trip request it responds to.

The operations of class TripShareApp specify how the DAs interact. This is shown in Fig. 2. First, the requestor (Bob, in this case) asks his local application to find a suitable trip by issuing a newTripRequest() event. The local application, using the framework, sends a newTripProposal() event to all his contacts. Those running the same application are able to process this event and respond to the request. In this case, the DAs of Ada and Cam return trip proposals because they match Bob's trip requirements. Bob then asks his local app to display all received proposals and selects one of them (in this case, Cam's), and asks his DA to confirm the trip. Should Cam decline the confirmation, Bob continues asking the rest of the proposers until one of the proposals is accepted (and hence Bob shares the trip with that contact), or all proposals are rejected (and Bob has to wait for the bus).

III. Trust Management in Digital Avatars

A. Modeling Digital Avatars

The initial architecture we propose for the DAs framework was described in [4]. We shall mention that communication between DAs is performed using the Siddhi CEP engine. We have extended its functionality to allow sending and receiving information between DAs.

To show how Digital Avatars applications can be specified in a high-level and platform-independent manner, we have developed a UML model of the framework, which is shown in Fig. 1. The central part of the class diagram presents a Digital Avatar (class DigitalAvatar). It contains a set of records (class DigitalAvatarRecord), and participates in a set of local applications (class Application), which represent the local instances of the general application when deployed in the local smartphones of the users. The DA records are used to store the persistent data managed by the different applications the DA participates in. In addition, records of class PersonalInfo represent information about other users, including their associated DA, name, phone, address and email. One of these records corresponds to the phone owner (related though role "me" with the DA). Groups of users can be defined and stored in records, too. Each DA record stores the last time it was updated (lastModification) and defines the level of privacy required to read or write it: public if any application can access it, granted if only certain users can access it, and private if the record can only be accessed by the DA. In case of a granted privacy level, readGroups and writeGroups define the groups of users that have permissions to read and write the record, respectively. For example, a DA can define that only physicians can write and read the records that contain the health information about the user, while the close family can only read them. Finally, class Clock is used to keep track of time, which is represented by an Integer value using the POSIX notation [22].

Now, if we have a look at Fig. 1, we can see the application of the Digital Avatars framework to our carpooling system. An application is modeled by class Application. A DA can participate in several local applications, i.e., those installed in the phone. Classes TripShareApp, Trip, TripProposal and TripRequest model our DA application (cf. Section II-E).

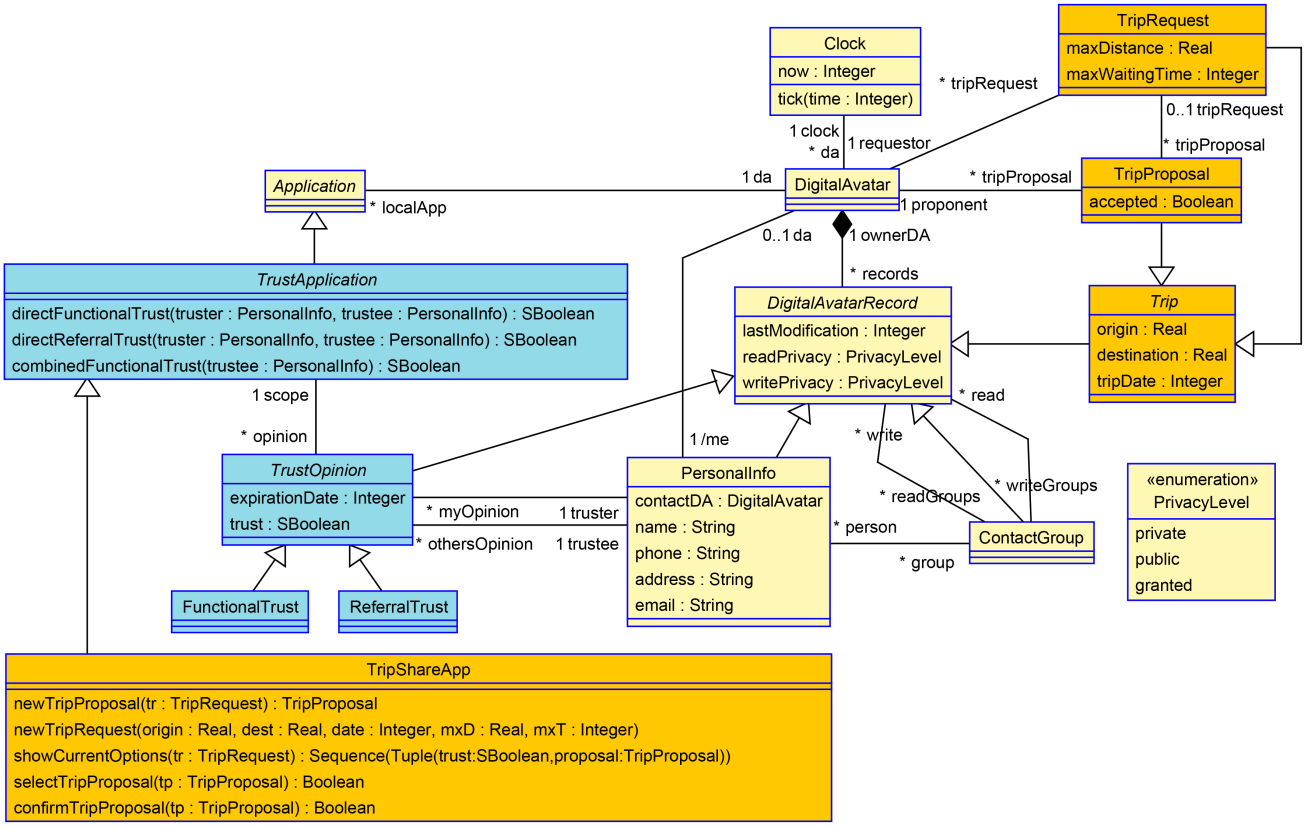

Figure 3: Class Diagram of the TripShare application incorporating trust concepts (shaded in blue color).

B. Modeling Trust

Note how at this level, the model abstracts away the details of how distributed communications between DAs take place, i.e., how methods are actually implemented in terms of CEP events that are passed between the corresponding CEP engines, or how the DA records are created, updated or deleted. When a DA wants to invoke a method, it creates an event with the appropriate information and passes it to its local CEP engine, which sends it to the intended recipients. These events include not only the source and destination DA identifiers, but also the identifier of the application itself, which provides the *scope* for the interaction. Upon arrival at the target engines, events are passed to the rules defined by the corresponding application, which processes them and performs the appropriate actions. Responses to the initial DA can also be generated as a result of this process, which will follow a similar communication path. For the purposes of this work all these details are not relevant and therefore we will abstract them away from our UML models, which remain at a higher level of detail.

Figure 3 shows a high-level model of our proposal for incorporating trust in DA applications. It adds four new classes to the model previously shown in Fig. 1. These new classes are shaded in blue in Fig. 3.

In the first place, abstract class TrustApplication extends class Application to represent those local applications able to deal with trust. Then, class TripShareApp now inherits from TrustApplication. Two new classes, FunctionalTrust and ReferralTrust, represent the degree of trust (*functional* and *referral*, resp.) that a truster has on a trustee. These two entities are modeled by the corresponding DA records of type PersonalInfo that represent the persons playing these roles. The common attributes of classes FunctionalTrust and ReferralTrust are stored in the abstract class TrustOpinion, which is just another DA record. Attribute trust represents the degree of trust, while attribute expirationDate sets a limit for the validity of the trust opinion (-1 if it never expires). Again, type Integer is used since time is represented in POSIX format. The scope of every opinion is given by the corresponding class TrustApplication, which provides three methods that allow computing the degree of trust that a truster has in a trustee in the scope defined by the application. The first two methods return the direct functional or referral trusts, respectively. They do that by simply looking for an opinion of such type within the scope of the application that specifies this information. If no record is found, they return the null value.

Method combinedFunctionalTrust() provides the main service of class TrustApplication. Using the DA of the owner as truster, it computes the degree of trust of a given trustee in the scope of the application. First, if there is a record that defines the direct functional trust of the truster on the trustee, this method returns such a value. If not, the method looks for those other records of people for whom the truster has a referral

Code 1: Assigning trust to a trustee.

```
combinedFunctionalTrust(trustee:PersonalInfo):SBoolean =
  let myFunctionalTrust : SBoolean =
     self.directFunctionalTrust(self.da.me,trustee) in
  -- if I have a functional trust, I return this value
  if myFunctionalTrust <>null then myFunctionalTrust
  else -- Do I know people who trust that person?
    let RT : Set(PersonalInfo) = -- contacts who have a
       -- functional trust about "trustee" in this app
      self.opinion
      ->select(o|o.oclIsTypeOf(FunctionalTrust) and
                o.trustee = trustee)
      ->collect(o|o.truster) -- from these, I select those
        -- for which I have a referral trust
      ->select(c |
          self.directReferralTrust(self.da.me,c)<>null)
      ->asSet() in
    if RT->isEmpty then -- no idea about that person
      SBoolean(0,0,1,0.5)
    else -- compute the direct opinions of referrers
      let opinions:Sequence(SBoolean) =
        RT->iterate(c ; s:Sequence(SBoolean) = Sequence{} |
        s->append(self.directFunctionalTrust(c,trustee).
        -- and discount their referral opinions
           discount(Sequence{self.
                directReferralTrust(self.da.me,c)}))) in
      let f:SBoolean = opinions->first() in
      let Q:Sequence(SBoolean) = opinions->excluding(f) in
      if Q->isEmpty() then f  -- only one opinion
      else f.aleatoryCumulativeBeliefFusion(Q) -- fuse them
      endif
    endif
  endif
```

trust, and who have a direct functional trust in the trustee. It then applies the discount operator on these indirect opinions, and fuses them using the Cumulative Fusion operator [7].

More precisely, Code 1 shows the OCL specification of the method that computes the trust of a person (trustee) by the truster (self.da.me) in the scope of an application (self).

Using this new model and the associated operations, the process is exactly the same as the one shown in Fig. 2, but now the operation showCurrentOptions() of class LolcalTripShareApp returns a sequence of proposals, together with their associated degree of trust. The following listing shows an example of the result of this operation, including the degree of trust associated to each proposal. Now Bob can make a much more informed decision about ordering the two trip proposals, and will ask Ada first, instead of Cam.

```
Sequence{
  Tuple{proposal = TripProposal2,
        trust = SBoolean(0.8,0.0,0.2,0.5)},
  Tuple{proposal = TripProposal1,
        trust = SBoolean(0.4,0.2,0.4,0.5)}
}
```

Note that we have used a base rate of 0.5 because we assume that we have no prior evidence about these users. More precise values could be derived if we were able to evaluate the users' reputation [7], or from estimations made using Machine Learning tecnhiques from previous experiences.

As for the discount and fusion operators, we have used the extended versions of those originally defined by Jøsang in [7]. In particular, the discount operator uses projection instead of belief as the discounting factor, which obtains more natural results [23], [24]. Similarly, the fusion operator is based on the improved version of the original one, as defined in [25]. The implementation of the extended types and their corresponding operations (including the discount and fusion operators) is available from our Github repository [26].

C. Implementation

This section describes the implementation that we have developed for the carpooling application TripShareApp.

First of all, we have to differentiate the three main components involved in any Digital Avatars application:

- The Digital Avatar itself, with its records of information and the local API for their management;
- the CEP engine, with a list of rule scripts running in parallel; and
- the specific third-party application, which defines apart from the rule scripts for CEP, its specific behavior for the information processing.

These three components are deployed in the smartphone, where all the data is handled, therefore ensuring the required levels of privacy and control by the user.

The DA records are stored as JSON documents using a CouchBase Lite NoSQL database.[1] They are managed and accessed via the DA API, which is accessible only locally on the smartphone. It handles all security and privacy issues, controlling the operations executed on the data and checking access permissions.

The CEP engine is the second component of the framework. It is also application-independent, and has been implemented using Siddhi's lightweight CEP engine [27] and its extensions for Android devices. It is capable of running in the background, always ready to receive events from the smartphone's sensors. The engine can also perform actions on the phone, for example, emitting a sound or displaying a notification.

The engine is able to run several scripts in parallel with their own rules and event sources. We have extended the functionality of Siddhi to support our own features. Specifically, we support communication between distributed CEP engines running on different smartphones by exchanging events. We use the OneSignal platform, which is based on Firebase Cloud Messaging. It provides an API service that allows smartphones to send and receive notifications. We integrated this service into the Siddhi engine by adding a new type for interfaces @source and @sink. These are the Siddhi interfaces for the different data sources and actuators. We developed the android-message type for both interfaces, using the OneSignal API for receiving push notifications or sending them.

The third component of the Digital Avatars framework is the applications. To illustrate how they are implemented, let us focus on the carpooling scenario, realized by means of the TripShareApp application. First, it stores its information (i.e. the user trips) using the DA records. To work with the CEP engine, the TripShareApp implements an Android Broadcast Receiver. This class is continuously listening for Android events with a specific action (i.e., identifier) that are sent as an endpoint of the CEP script. This is illustrated in the script

[1] https://www.couchbase.com/products/lite

excerpt in Code 2, which shows the code to be executed when a trip query arrives at the CEP engine of one of Bob's contacts.

A @source of type android-message (line 1) receives the external message with the query. This is the source that is listening using the OneSignal platform. It specifies an app identifier (line 2) that allows the CEP engine to identify the application script that should handle the event. The attributes of this message are defined in lines 5-13. They are mapped to the attributes of the event that is going to be transmitted through the receiveTripQuery stream in line 14. In this example, Bob requests to take a trip from origin to destination, specifying the date and time he wants to take the trip. Bob also defines the maximum distance he is willing to deviate from his origin and destination points, as well as the maximum waiting time before departure. The OneSignalId indicates the identifier of the sender, so that respondents can reply to it.

Once the external message is captured, it is passed to the application using a @sink of type android-broadcast (line 24). The stream (line 27) defines the event that the application will receive. The connection between the source and the sink is established in line 37. The receiver of the TripShareApp is always listening for events whose identifier is TripShare_TripQuery (as defined in the @sink) and will invoke the corresponding operation as soon as the event is received.

Sending messages from the contact's application to Bob's CEP engine follows a similar process. Thus, to respond to the trip proposal, the application places the response message in the CEP engine using a @source of type android-broadcast, which is connected to a @sink of type android-message, specifying the requester as recipient of the message.

These communication mechanisms enable the applications running in the smartphones to exchange information, request services and respond to them. The behavior of the application can be written in Java to implement the required functionality.

To implement trust management, on top of the basic functionality, we proceed as specified in Sect. III-B. Thus, every trip proposal is endowed with the trust that the requester assigns to the proposer, using the algorithm described in Code 1. For this, the DA uses the records that determine the *functional trust* on the trustee or, in case this information does not exist, the TripShareApp uses the API to consult the DA for *functional trust* of all the personal contacts for which it has a referral trust in the context of the application. If there are several of them, the DA combines them using the aleatory cumulative belief fusion (ACBF) operator [7].

Then, every DA needs to store and manage a set of trust records (see Fig. 3). The TrustOpinion records that specify the phone owner's functional trust over these contacts are defined when the user configures the application, as it provides the scope of such trust opinions. Similarly, the phone owner needs to specify the ReferralTrust records with their *referral trust* over their contacts in the scope of the application. A third set of records is also required: the functional trust over the trip proposer of each of the owner's contacts for whom a referral trust is defined. Two strategies are possible to obtain this information. Upon receipt of a trip proposal, the DA

Code 2: Excerpt of Siddhi CEP script for the TripShareApp.

```
1  @source(type='android-message',
2      appid ='TripShare_TripQuery',
3      @map(type='keyvalue',
4           fail.on.missing.attribute='false',
5           @attributes(
6              originLatitude='originLatitude',
7              originLongitude='originLongitude',
8              destinationLatitude='destinationLatitude',
9              destinationLongitude='destinationLongitude',
10             date='date',
11             time='time',
12             maxDistance='maxDistance',
13             waitingTime='waitingTime',
14             onesignalId='sender')))
15 define stream receiveTripQuery(
16             originLatitude double,
17             originLongitude double,
18             destinationLatitude double,
19             destinationLongitude double,
20             date String,
21             time String,
22             maxDistance double,
23             waitingTime double,
24             onesignalId String);
25 @sink(type='android-broadcast',
26     identifier='TripShare_TripQuery',
27     @map(type='keyvalue'))
28 define stream tripQuery(
29             originLatitude double,
30             originLongitude double,
31             destinationLatitude double,
32             destinationLongitude double,
33             date String,
34             time String,
35             maxDistance double,
36             waitingTime double,
37             onesignalId String);
38 from receiveTripQuery select * insert into tripQuery;
```

sends a query to all the contacts for whom it has defined a *referral trust*, asking them for their functional trust on the trip proposer. Once the replies to these queries are received, the trust is computed. Alternatively, a separate process running in background is in charge of asking these queries to the phone owner's contacts for whom a referral trust is defined, in case this information is not available in the DA, or it has expired. Using this approach, the application can readily use the available DA records, because it can assume the information is always updated. This is the approach we currently follow, since it allows more efficient trust management and calculation.

The complete code of the example application, along with the current version of the Digital Avatars framework, can be found on GitHub [28]. The implementation uses the Java library with the datatypes extended with uncertainty and in particular datatype SBoolean, which represents trust opinions and its operations [26].

IV. Related Work

Since the 90's, the scientific community has shown an enormous interest in the study of the ideas underlying the concept of trust [29], [30]. Experiments have been conducted in different areas like electronic commerce, information systems, or other disciplines such as psychology, economics or security. In this section we will review some of these works. We have organized it according to three main dimensions: the

trust model used, i.e., how trust is represented and calculated; how Subjective logic is used for representing trust; and how trust is modeled in social computing applications.

A. Trust model

The different proposals vary from each other in a number of important points, with the most relevant being the underlying trust model, i.e., how they compute trust, the inputs they accept and the produced outputs (binary, discrete or real values and linguistic labels). Proposals also vary in the algorithms employed to compute trust and to detect false information (simple heuristics, learning and stochastic models).

Trust models can be classified as either quantitative or qualitative. In quantitative models, users provide numeric inputs and models estimate and convey trust in the same terms. In qualitative models, the exchanged information is not numerical, but usually binary or ordinal.

Jelenc and Trček [31], [32] propose both a formal system and a simulation framework based on qualitative and ordinal values. Their model allows estimating the most representative qualitative value on the basis of past evaluations, judgement of received opinions and degrees of trust in trust estimations.

Along the same line, using linguistic values with 3 and 5 level qualitative scales, Abdul-Rahman [33] proposes a model for determining agents' trustworthiness based on statistics collected by the agent on both direct experiences and recommendations from other agents. Agents do not maintain a database of specific trust statements in the form of "a trusts b with respect to context c". Instead, at a given point in time, the trustworthiness of a particular agent is obtained by summarizing the relevant subset of recorded experiences. The main advantage of these kinds of models is that they are more intuitive to humans. For example, a probability is generally more difficult to interpret than a qualitative value. However, quantitative models estimate trust with finer granularity, often exceeding the accuracy of qualitative models [34]. This is why in our proposal we decided to use a quantitative model, based on subjective opinions.

Among the works that use probability theory for the representation of trust we find the work of Travos [35]. Trust is calculated using past interactions between the agents. When no such interactions exist, the model uses reputational information obtained from third parties, which maybe inaccurate.

Yu et al. [36] also use a probabilistic approach to derive a trust model for rating the quality of certain services. Most proposals that apply polling algorithms to extract ratings needed for trust calculation are based on the Gnutella protocol [37] where each user queries its neighbors. In that sense, Yu's proposal improves those strategies by sending queries to a calculated subset of neighbors whose credibility is maximized. To avoid values that may be noisy or unreliable, Yu labels users' ratings as complementary, exaggerated positive, and exaggerated negative, and tries to identify misleading and unreliable peers.

Our work differs from these types of works in two main aspects. First, we use subjective opinions instead of probabilities in order to incorporate the agents' uncertainty into the decision process. Second, these trust models are mostly based on polls and reputations, which is not our case. In our work, both functional and referral trusts are individually decided by the phone user because of the individual nature of digital avatars and the explicit intention of our approach to avoid centralized information or that coming from external untrusted sources.

B. Trust computing based on subjective logic

Unlike other proposals that considered trust as a single value, e.g., a probability, Jøsang [7], [38], [39] used subjective opinions that enabled the incorporation of two essential features of trust: subjectivity and uncertainty. Additional operators were defined to propagate and combine trust, namely discounting and opinion fusion.

The original works by Jøsang were later extended by different authors. For example, improved versions of the fusion operators were defined in [25] in order to deal with some specific situations not correctly covered in the original versions. Similarly, the discount operator was also improved in [23] to produce more natural results. The specification and Java implementations that we have developed for the Digital Avatar Framework are based on these latter works.

Some authors have proposed other trust models based on Subjective logic. For example, Feng et al. [24] have extended Jøsang's trust model to situations where trust is not only asymmetric but mutual, can dynamically change, and is influenced by external events. They also used the discount operator defined in [25]. Feng's model is based on a previous work [40] that uses a Bayesian model instead of Subjective logic and introduces a sliding time window to update trust values. Some of these mechanisms (such as time windows, given by the records' expiration date) are present in our model, although we deviate significantly from these works because in our case, trust is asymmetric but may not be mutual due to the individual nature of the decisions made by the Digital Avatar owner. In this sense, our model of trust is closer to Jøsang's but using the improved versions of his original operators.

C. Trust in social computing

The Internet of Things (IoT) paradigm has driven the creation of complex networks of users where they often cooperate and coordinate, using their mobile devices, to carry out collaborative social computing. In this context, the need to infer the degree of trust between users plays a crucial role in carrying out social collaboration. Hence, many works have attempted to discover relationships between entities with social trust models.

Chen et al. [41] propose a global system for building hierarchical trust models for mobile social networks. Given two users, the approach is able to calculate the best communication path between them by providing the one that maximizes trust. They propose a clustering algorithm to produce a hierarchy of clusters and groups. Trust is calculated by taking into account the trust value of the group, the level of contacts, the evolution of the interaction and the user attributes. However, the trust

they propose is a static trust (based on user attributes) that does not take into account an indispensable consideration: the scope.

The trust model proposed by Li et al. [6] defines three factors: the similarity between user profiles, the reputation of the users, and the history of common friends. They also work with User Profiles, similar to what we call Digital Avatars, and compute the semantic distance between profiles to find their similarity. However, the trust they calculate is based on a set of quantifiable facts that both users have in common: similarities in the profile, friends in common, and so on. Trust is in itself a much more subjective aspect, as we note in our proposal, and even if two subjects have many properties in common, their trust can be very low based on shared experiences.

Ceolin and Potenza [42] propose a framework for estimating trust in social networks, based on user demographics, knowledge and network centrality. They use subjective logic, as we do, although they calculate their trust estimates using a global view of the social network, whereas we rely only on local (individual) information.

In his work, Golbeck [43] developed two implementations for computing trust in the context of social networks: one for binary-valued trust networks (i.e., trust-no trust) and one for networks that assigns continuous values to trust, in the range [0,1], called TidalTrust. In TidalTrust, trust is computed by performing a recursive search following the paths connecting people in the network and the trust values associated with each of those links. The depth of the search can be bounded for efficiency reasons. This approach can be seen as similar to the Trust networks defined by Jøsang [7]. Currently, we use just one level because our initial experiments showed that trust decreased rapidly when traversing the network of contacts beyond neighbors of neighbors. Anyway, we need to further investigate this issue.

Trust has also been extensively studied in collaborative P2P environments. In most cases, the concept of reputation [44]–[46] is used for its computation. Reputation has been defined as "the common opinion that people have about someone or something: the way in which people think of someone or something" [7]. Thus, reputation is a quantity derived from the underlying social network, which is globally visible to all members of the network. Reputation can be thought of as a collective measure of trust based on the referrals or ratings from members in a community. Note that the collective (*global*) nature of reputation differs from that of *individual* trust, and therefore its definition and management in peer-to-peer social computing applications requires different mechanisms than those described in this paper. Here we have tackled the representation and management of trust from the point of view of individual digital avatars, given the decentralized nature of our proposal and the explicit intention to avoid any kind of centralized global information. Incorporating reputation to our model is part of our future work.

Social Net [47] infers shared interests between people by storing the IDs of the nearby devices and analyzing over a long period of time their proximity. However, the relationship between geographic proximity and trust may be entirely casual, rather than causal. The fact that you share nearby locations with some people repeatedly over time does not imply that you have more trust in them.

V. Conclusions

This contribution proposes a trust management system for collaborative social computing applications. It builds on the Digital Avatars framework, extending it with the explicit representation and management of trust information about both service providers (*functional trust*) and other users' opinions about these service providers (*referral trust*) in peer-to-peer environments.

We have specified the proposal using high-level, platform-independent UML models, and provided the OCL specification of trust management operations. As a proof of concept of the proposed approach, and to demonstrate the proposal, we have developed an implementation of a mobile-based application that takes into account trust opinions for making informed decisions.

This work can be continued in several directions. First, we plan to evaluate our proposal with more applications to better understand and appraise its advantages and limitations. One domain of particular interest is healthcare, where there are elements of patient trust in clinicians and also the use of digital health records to support direct care delivery or even secondary research activities. This could also open up some questions about how to model the evolution of trust in both clinical service providers and IT solution providers. Second, we want to conduct dedicated empirical experiments with real users to evaluate its usability and usefulness. As mentioned above, the implementation of a reputation management system in peer-to-peer environments may require further research, due to its global nature. Thus, combining global reputation with individual trust in the context of collaborative social applications is another line of research we want to explore. For example, this could allow us to more precisely define the base rate of subjective opinions that represent individual trust. Incorporating our trust model to other social computing application frameworks, such as [48]–[51] could be an interesting line of research, too. Finally, we would like to extend this approach with the use of Machine Learning techniques that can automatically modify users' trust based on their past experiences and the quality of service they obtain.

Acknowledgment

We would like to thank the reviewers for their insightful comments and constructive suggestions, which have significantly helped us to improve the paper. This work was funded by the Spanish Research Projects PGC2018-094905-B-100 and RTI2018-098780-B- I00.

References

[1] J. Guillén, J. Miranda, J. Berrocal, J. García-Alonso, J. M. Murillo, and C. Canal, "People as a service: A mobile-centric model for providing collective sociological profiles," *IEEE Software*, vol. 31, no. 2, pp. 48–53, 2014.

[2] J. Miranda, N. Mäkitalo, J. García-Alonso, J. Berrocal, T. Mikkonen, C. Canal, and J. M. Murillo, "From the internet of things to the internet of people," *IEEE Internet Computing*, vol. 19, no. 2, pp. 40–47, 2015.

[3] A. Pérez-Vereda and C. Canal, "A people-oriented paradigm for smart cities," in *Proc. of ICWE'17*, ser. LNCS, vol. 10360. Springer, 2017, pp. 584–591.

[4] M. F. Bertoa, N. Moreno, A. Pérez-Vereda, D. Bandera, J. M. Álvarez-Palomo, and C. Canal, "Digital avatars: Promoting independent living for older adults," *Wirel. Commun. Mob. Comput.*, vol. 2020, pp. 8 891 002:1–8 891 002:11, 2020.

[5] J. Troya, N. Moreno, M. F. Bertoa, and A. Vallecillo, "Uncertainty representation in software models: A survey," *Software and Systems Modeling*, 2021. [Online]. Available: https://doi.org/10.1007/s10270-020-00832-3

[6] J. Li, Z. Zhang, and W. Zhang, "Mobitrust: Trust management system in mobile social computing," in *Proc. of CIT'10*. IEEE Computer Society, 2010, pp. 954–959.

[7] A. Jøsang, *Subjective Logic - A Formalism for Reasoning Under Uncertainty*. Springer, 2016.

[8] O. Etzion and P. Niblett, *Event Processing in Action*. Manning Publications, 2010.

[9] D. Gambetta, "Can we trust trust?" in *Trust: Making and Breaking Cooperative Relations*. University of Oxford, 2000, pp. 213–237.

[10] L. Burgueño, M. F. Bertoa, N. Moreno, and A. Vallecillo, "Expressing confidence in models and in model transformation elements," in *Proc. of MODELS'18*. ACM, 2018, pp. 57–66.

[11] D. de Siqueira Braga, M. Niemann, B. Hellingrath, and F. B. de Lima Neto, "Survey on computational trust and reputation models," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 101:1–101:40, 2019.

[12] T. Grandison and M. Sloman, "A survey of trust in internet applications," *IEEE Commun. Surv. Tutorials*, vol. 3, no. 4, pp. 2–16, 2000.

[13] D. H. McKnight and N. L. Chervany, "Conceptualizing trust: A typology and e-commerce customer relationships model," in *Proc. of HICSS-34*, 2001.

[14] B. Adams and R. Webb, "Model of trust development in small teams," Department of National Defense, Tech. Rep. CR 2003-016, 2003.

[15] D. Griffin and A. Tversky, "The weighing of evidence and the determinants of confidence," *Cognitive Psychology*, vol. 24, no. 3, pp. 411–435, 1992.

[16] W. Petrusic and J. Baranski, "Judging confidence influences decision processing in comparative judgments," *Psychonomic Bulletin & Review*, vol. 10, p. 177–183, 2003.

[17] JCGM 100:2008, *Evaluation of measurement data – Guide to the expression of uncertainty in measurement (GUM)*, Joint Committee for Guides in Metrology, 2008, http://www.bipm.org/utils/common/documents/jcgm/JCGM_100_2008_E.pdf.

[18] M. F. Bertoa, L. Burgueño, N. Moreno, and A. Vallecillo, "Incorporating measurement uncertainty into OCL/UML primitive datatypes," *Softw. Syst. Model.*, vol. 19, no. 5, pp. 1163–1189, 2020.

[19] P. Muñoz, L. Burgueño, V. Ortiz, and A. Vallecillo, "Extending OCL with Subjective Logic," *Journal of Object Technology*, vol. 19, no. 3, pp. 3:1–15, Oct. 2020.

[20] W. Feller, *An Introduction to Probability Theory and Its Applications*. Wiley, 2008.

[21] B. de Finetti, *Theory of Probability: A critical introductory treatment*. John Wiley & Sons, 2017.

[22] IEEE Std 1003.1-2008, *The Open Group Base Specifications. Issue 7, Sect. 4.16, Seconds Since the Epoch*, 2016.

[23] H. A. Kurdi, B. Alshayban, L. Altoaimy, and S. Alsalamah, "Trustyfeer: A subjective logic trust model for smart city peer-to-peer federated clouds," *Wirel. Commun. Mob. Comput.*, vol. 2018, 2018.

[24] J. Tian, J. Zhang, Z. Peipei, and X. Ma, "Dynamic trust model based on extended subjective logic," *KSII Trans. Internet Inf. Syst.*, vol. 12, no. 8, pp. 3926–3945, 2018.

[25] R. W. van der Heijden, H. Kopp, and F. Kargl, "Multi-source fusion operations in subjective logic," in *Proc. of FUSION'18*. IEEE, 2018, pp. 1990–1997.

[26] Atenea Research Group, "Uncertain datatypes – Git repository," https://github.com/atenearesearchgroup/uncertainty, 2021.

[27] S. Suhothayan, K. Gajasinghe, I. L. Narangoda, S. Chaturanga, S. Perera, and V. Nanayakkara, "Siddhi: a second look at complex event processing architectures," in *Proc. of SC@GCE'11*. ACM, 2011, pp. 43–50.

[28] A. Pérez-vereda, "Digital avatars: Trip share application – Git repository," https://github.com/apvereda/Digital-Avatars-TripShare, 2021.

[29] G. C. M. Amaral, T. P. Sales, G. Guizzardi, and D. Porello, "Towards a reference ontology of trust," in *Proc. of OTM'19*, ser. LNCS, vol. 11877. Springer, 2019, pp. 3–21.

[30] G. C. M. Amaral, R. S. S. Guizzardi, G. Guizzardi, and J. Mylopoulos, "Ontology-based modeling and analysis of trustworthiness requirements: Preliminary results," in *Proc. of ER'20*, ser. LNCS, vol. 12400. Springer, 2020, pp. 342–352.

[31] D. Jelenc and D. Trček, "Qualitative trust model with a configurable method to aggregate ordinal data," *Auton. Agents Multi Agent Syst.*, vol. 28, no. 5, pp. 805–835, 2014.

[32] D. Trček, "Computational trust management, QAD, and its applications," *Informatica*, vol. 25, no. 1, pp. 139–154, 2014.

[33] A. Abdul-Rahman and S. Hailes, "Supporting trust in virtual communities," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, 2000, pp. 9 pp. vol.1–.

[34] A. Tversky and D. Kahneman, "Judgment under uncertainty: Heuristics and biases," *Science*, vol. 185, no. 4157, pp. 1124–1131, Sep. 1974. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/17835457

[35] W. T. L. Teacy, J. Patel, N. R. Jennings, and M. Luck, "Travos: Trust and reputation in the context of inaccurate information sources," *Autonomous Agents and Multi-Agent Systems*, vol. 12, no. 2, pp. 183–198, March 2006. [Online]. Available: https://eprints.soton.ac.uk/262129/

[36] B. Yu, M. Singh, and K. Sycara, "Developing trust in large-scale peer-to-peer systems," in *IEEE First Symposium onMulti-Agent Security and Survivability, 2004*, 2004, pp. 1–10.

[37] D. Han and J. Zhang, "An optimized gnutella-like P2P protocol in mobile networks," *J. Networks*, vol. 7, no. 9, pp. 1464–1471, 2012.

[38] Z. Milosevic, A. Jøsang, T. Dimitrakos, and M. A. Patton, "Discretionary enforcement of electronic contracts," in *Proc. of EDOC'02*. IEEE Computer Society, 2002, pp. 39–50.

[39] T. Dimitrakos, I. Djordjevic, Z. Milosevic, A. Jøsang, and C. I. Phillips, "Contract performance assessment for secure and dynamic virtual collaborations," in *Proc. of EDOC'03*. IEEE Computer Society, 2003, pp. 62–75.

[40] R. Feng, X. Han, Q. Liu, and N. Yu, "A credible bayesian-based trust management scheme for wireless sensor networks," *Int. J. Distrib. Sen. Netw.*, vol. 2015, Jan. 2015.

[41] S. Chen, G. Wang, and W. Jia, "Cluster-group based trusted computing for mobile social networks using implicit social behavioral graph," *Future Generation Computer Systems*, vol. 55, pp. 391–400, 2016.

[42] D. Ceolin and S. Potenza, "Social network analysis for trust prediction," in *Proc. of IFIPTM'17*, ser. IFIP Advances in Information and Communication Technology, vol. 505. Springer, 2017, pp. 49–56.

[43] J. A. Golbeck, "Computing and applying trust in web-based social networks," Ph.D. dissertation, University of Maryland, 2005. [Online]. Available: http://hdl.handle.net/1903/2384

[44] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in P2P networks," in *Proc. of WWW'03*. ACM, 2003, pp. 640–651.

[45] E. Damiani, S. Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," in *Proc. of CCS'02*. ACM, 2002, pp. 207–216.

[46] P. Michiardi and R. Molva, "CORE: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *Proc. of IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security*, ser. IFIP Conference Proceedings, vol. 228. Kluwer, 2002, pp. 107–121.

[47] M. A. Terry, E. D. Mynatt, K. Ryall, and D. Leigh, "Social net: using patterns of physical proximity over time to infer shared interests," in *Extended abstracts of CHI'02*. ACM, 2002, pp. 816–817.

[48] H. Mao, M. Xiao, A. Liu, J. Li, and Y. Hu, "OCC: opportunistic crowd computing in mobile social networks," in *Proc. of DASFAA'16*, ser. LNCS, vol. 9645. Springer, 2016, pp. 254–267.

[49] J. Bajo, A. T. Campbell, and X. Zhou, "Mobile sensing agents for social computing environments," in *Proc. of PAAMS'16*, ser. AISC, vol. 473. Springer, 2016, pp. 157–167.

[50] S. Mohan, N. Agarwal, and L. Al-Doski, "Mobile network-aware social computing applications: a framework, architecture, and analysis," *J. Ambient Intell. Humaniz. Comput.*, vol. 4, no. 1, pp. 43–56, 2013.

[51] H. M. Tran, K. V. Huynh, K. D. Vo, and S. T. Le, "Mobile peer-to-peer approach for social computing services in distributed environment," in *Proc. of SoICT'13*. ACM, 2013, pp. 227–233.