Managing Reputation in Collaborative Social Computing Applications

Nathalie Moreno[†], **Alejandro Pérez-Vereda**[*], **and Antonio Vallecillo**[†]
[†]ITIS Software. Universidad de Málaga, Spain
[*]Universidad de Castilla-La Mancha, Spain

ABSTRACT Reputation is a fundamental concept for making decisions about service providers. However, managing reputation in peer-to-peer distributed applications is not easy due to the lack of a central server that can compute this property from user opinions. Moreover, users have to marry this information with their individual trust in the service provider, which may be based on their past experiences, the opinions of their direct contacts, or both. This paper develops a reputation management system embedded in the Digital Avatars framework for collaborative social computing applications, using subjective logic. We show how the reputation of a given service provider can be calculated using the users' opinions about it, and how this reputation can be explicitly represented, managed and combined with the trust that individual service requesters may have in them, in order to make better informed decisions.

KEYWORDS Collaborative social computing applications, Trust, Reputation, Subjective Logic.

1. Introduction

Social computing (SC) is the discipline that aims at designing and operating digital systems that support useful functionality by making socially produced information available to their users. The information is not anonymous because it is linked to people, who are in turn linked to other people (Schuler 1994). SC is gaining acceptance for developing social applications, since they allow humans to become the main protagonists of these applications, not only as beneficiaries but also as active players.

Most social applications, such as those available from Google, Facebook, Amazon, or any of the big IT players, follow a server-centric model where the content created by distributed users is transferred to cloud servers. In these applications, citizens are required to hand over their personal information to the service providers, hence losing control of their data, which becomes the property of third parties. Once under their control, this information can be employed for any purpose, sometimes different from those it was generated for by the citizens.

JOT reference format:
Nathalie Moreno, Alejandro Pérez-Vereda, and Antonio Vallecillo. *Managing Reputation in: Collaborative Social Computing Applications*. Journal of Object Technology. Vol. 21, No. 3, 2022. Licensed under Attribution 4.0 International (CC BY 4.0) http://dx.doi.org/10.5381/jot.2022.21.3.a1

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.

However, the fully distributed peer-to-peer nature of these systems introduces some difficulties for dealing with concepts which are inherently global. In particular, *reputation* is a fundamental concept for making decisions about service providers (e.g., a restaurant, a hairdresser, or a transportation company) in social computing applications. Reputation is normally computed by aggregating the opinions of individual users about a given entity, such as a person or company providing a service. However, in MCSC applications there is no central authority to calculate such a reputation, nor is there a central authority for users to consult it. In addition, combining the global reputation of a given provider with the users' individual trust in it is not an easy task.

This paper develops a reputation management system embedded in the Digital Avatars framework (Bertoa, Moreno, et al. 2020) for collaborative social computing applications. We show how the reputation of a service provider can be calculated using the users' opinions about it, and how this reputation can be explicitly represented, managed and combined with the trust that an individual service requester may have in them, in order to make better informed decisions. More precisely, this paper aims at answering the following research questions:

Q1. How reputation can be expressed in MCSC applications for its effective representation, operation and management?
Q2. How the reputation of a given entity of a MCSC application can be computed and updated in such peer-to-peer and decentralised contexts?
Q3. When a user is trying to decide on a service, how to marry the reputation of the service with the individual opinion (i.e., trust) that the user has about it?

The proposal is specified using high-level models that can also be simulated and validated, and can be used as a guide for the corresponding implementations in an existing social computing application platform.

This paper builds on our previous work on the representation and management of trust in collaborative social computing applications (Muñoz et al. 2021), where we extended the initial Digital Avatar framework (Bertoa, Moreno, et al. 2020) with trust. The solution presented in this paper incorporates the concept of reputation, and has been successfully applied in a collaborative decision-making system where users need to select a specialist physician from a list of practitioners. That system will be used in this paper to illustrate our proposal.

The structure of this document is as follows. After this introduction, Section 2 briefly describes the background of our work and presents the example that is used to motivate it. Then, sections 3, 4 and 5 aim at responding the three research questions posed above, and Section 6 describes our proposal. Finally, Section 7 relates our work to other similar approaches and Section 8 concludes with an outline of future work.

2. Context and Definitions

To set the paper terminology, this section briefly describes the context of the work and the main concepts used in the paper.

2.1. 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, service providers or, in general terms, to *agents*, to perform an action (*functional* trust) or to report about the reliability of other people or agent (*referral* trust) (Gambetta 2000; Jøsang 2016). 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 (Burgueño et al. 2018)—see Sect. 2.2 below.

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 (de Siqueira Braga et al. 2019). 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 (Grandison & Sloman 2000).

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 cook. Thus, trust is not absolute but must be specified within a *scope* (Jøsang 2016; Grandison & Sloman 2000).

Finally, trust in people is *subjective*, i.e., it depends on the truster, and is normally conditioned by uncertain factors (McKnight & Chervany 2001). As stated in (Adams & Webb 2003): "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."

2.2. 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* (Burgueño et al. 2018; Griffin & Tversky 1992; Petrusic & Baranski 2003), 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" (JCGM 100:2008 2008).

From a generic decision-making perspective, confidence is the degree of belief in a given hypothesis (Griffin & Tversky 1992) 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.

The treatment of confidence in software models was already described in some of our previous works (Burgueño et al. 2018; Bertoa, Burgueño, et al. 2020; Muñoz et al. 2020; Troya et al. 2021; Burgueño et al. 2022), and therefore we will not consider it further in this paper.

2.3. Reputation

A concept closely related to trust is *Reputation*, which has been defined as "the common opinion that people have about someone or something: the way in which people think of someone or something" (Jøsang 2016). 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 measure of collective trust based on the referrals or ratings from members in a community. Like trust, reputation is not absolute, but is defined within a *scope*: I can have a decent reputation as a cook, but a lousy reputation as a driver.

Note that the collective (*global*) nature of reputation differs from the *individual* nature of trust. Therefore, defining, computing, and managing reputation in SC applications require different mechanisms than those used to represent, compute, and manage individual trusts.

2.4. A motivating example

Trust and reputation are two sides of the same coin. Decision making in collaborative social computing applications needs not only the concept of trust, but also that of reputation. To illustrate this claim, suppose, e.g., that Alberto is a stressed businessman who wakes up one morning with a suspicious chest pain and needs to consult a good cardiologist. Alberto has private insurance and uses an app on his cell phone that allows him to choose from a number of specialists located in his area. He has never needed a cardiologist, so he has no direct references of any of them. So, he consults his app to find out the reputation of these specialists and, based on that, initially selects the one with the best *reputation*. In addition, as any of us would do, he decides to ask his direct contacts for references to a good specialist on his list (i.e., their *trust* in them) and with all the information gathered (reputation and trust) he can make a better informed decision.

This is just one of the many cases in which reputation and trust need to be combined in decision-making. However, such a combination is not that simple. What happens if some of Alberto's contacts have had negative experiences with his first choice? Normally, Alberto would rank his choices differently after considering his contacts' trust. However, how to combine reputation and trust? How should Alberto weight them in his final decision?

3. Representing Reputation in MCSC Apps

To represent reputation we will use Subjective Logic (Jøsang 2016), 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 combine individual opinions for computing reputation, and then to marry it with the users' trust.

To specify, design and develop MCSC applications we will use the Digital Avatars collaborative framework (Bertoa, Moreno, et al. 2020). This framework is based on the People-as-a-Service (PeaaS) model (Guillén et al. 2014), 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) (Miranda et al. 2015), smart cities (Pérez-Vereda & Canal 2017) or gerontology (Bertoa, Moreno, et al. 2020).

This section briefly introduces Subjective Logic (Sect. 3.1), the Digital Avatars framework (Sect. 3.2) and our proposal for representing reputation in this context (Sect. 3.3).

3.1. Subjective logic

Traditionally, degrees of trust or confidence have been modeled using probabilities (i.e. numbers between 0 and 1), and reasoning about trust has been accomplished using probability theory (Feller 2008; de Finetti 2017). However, this approach falls short 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, rather than assigning x a confidence of 0.5, it might be preferable to say "I don't know." In general, forcing users to set probabilities to express their opinions could lead to unreliable conclusions (Muñoz et al. 2020). This is when Subjective logic can be of great help.

Subjective logic, introduced by Audun Jøsang (Jøsang 2016), 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 without any previous evidence.

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, 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, as it happens in our context.

To represent and operate with Subjective logic values in UML and OCL models, in (Muñoz et al. 2020) we defined an extension of their primitive datatype `Boolean`. The extended datatype, called `SBoolean`, provides a set of operators that can be used for logical reasoning with uncertain propositions. 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 p representing a confidence into type `SBoolean` is achieved by assigning the opinion $w_x = (p, 1 - p, 0, p)$ to x. Analogously, the *projection* p of an opinion w is a Real number in the range [0..1] that projects the opinion into a probability: $p = b + a * u$. Examples of the use and application of Subjective logic in models represented with UML/OCL can be found in (Muñoz et al. 2020).

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

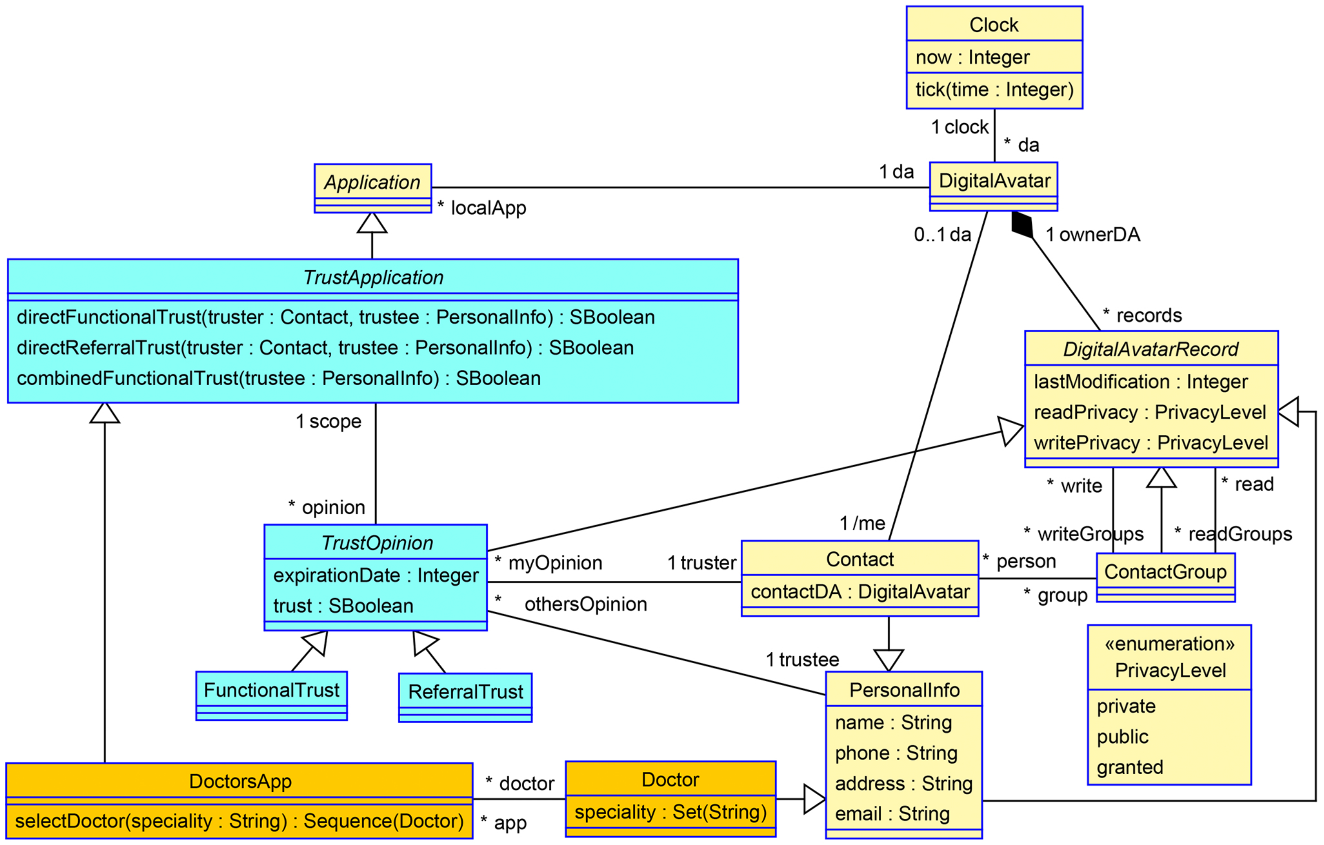


Figure 1 Class Diagram of the `Doctors` application incorporating trust concepts (shaded in blue color).

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. Specifically, Subjective Logic provides the following fusion operators:

- The Belief Constraint Fusion (BCF) operator is suitable in those contexts in which each agent has its own opinion and is willing to stick to it even if it means not reaching a consensus agreement. When faced with totally contradictory opinions, this operator returns complete a *vacuous* opinion, i.e., $(0,0,1,0.5)$.
- The Consensus & Compromise Fusion (CCF) operator is similar to BCF, but conflicting opinions are transformed into vagueness, i.e. uncertain opinions, to allow reaching a consensus.
- The Cumulative Belief Fusion (CBF) operator is applied when the opinions collected from different subjects are independent, i.e., they observed the fact in different moments, or from different perspectives. There are two variants of this operator, depending on the nature of the uncertainty: Aleatory (e.g. decisions based on tossing a coin) or Epistemic (due to the lack of knowledge, e.g., the opinions of different witnesses about who killed Kennedy).
- The Averaging Belief Fusion (ABF) operator is used when there are dependencies between evidences. Even when observing the same fact, each subject has formed his own opinion (e.g., the members of a jury).
- The Weighted Belief Fusion (WBF) is similar to the previous one, but assigns more weight to those opinions with less uncertainty, i.e., the more confidence in the opinion, the stronger the weight.

The fusion operator to use depends on the specific situation and the personal circumstances of the belief agents.

3.2. Digital Avatars

Having a mechanism that allows us to compute (or derive) the trust that a subject A can place in a subject B who offers a service can help us to make important decisions in the domain of Collaborative Social Computing Applications. In a previous work we proposed and implemented an architecture for managing such trust using what we called Digital Avatars (DA), see Fig. 1. A DA is an entity that resides on a person's phone or tablet and stores information about the user and their activities, while providing services to interact with other users' DAs.

Applications running locally on the mobile device query and update the information stored in the DA. In our proposal, these applications are also capable of handling trust. In the execution context of trust-managing applications (`TrustApplication`, in Fig. 1), the user issues a set of opinions that model the degree of trust (either functional or referral) that a truster has in a trustee. Trust is stored in the system as a Subjective opinion, associated with an expiration time after which the trust is no longer valid.

A TrustApplication implements two methods for obtaining the direct functional trust or the referral trust that a truster has on a trustee. To do so, they consult the local opinions in the context of that application stored in the DA, and return the specific TrustOpinion record, or a null value in case such information is not available. In addition to these two methods, a TrustApplication implements a method called combinedFunctionalTrust() to compute the trust on a given trustee. It operates as follows: if there is a record that defines the direct trust that the truster has in the trustee, the method returns that value. If not, the method searches for those contacts of the truster who have a trust opinion on the trustee, and merges their opinions. To merge these opinions we use the Epistemic Cumulative Belief Fusion operator (see previous section) because the individual opinions representing trust are normally obtained at different times or from different perspectives, and are epistemic in nature.

Figure 1 also shows the specification of the aforementioned application of selecting a good specialist in our DA framework, using the Unified Modelling Language (UML). The DoctorsApp application (shaded in orange) implements a service that, given a speciality, returns a list of doctors of that speciality, sorted according to the trust that the user has on them. Listing 1 shows the specification of that operation in OCL. The UML and OCL specifications in this article have been developed in USE (UML-based specification environment (Richters & Gogolla 1998, 2000)). USE is a system for the specification of information systems based on a subset of UML and OCL. In addition, USE provides an executable language called SOIL (Büttner & Gogolla 2014), which extends OCL to enable the simulation of UML systems.

```
selectDoctor(speciality:String) : Sequence(Doctor) =
  let l:Sequence(Doctor) = self.doctor->
          select(d | d.speciality->includes(speciality))
            ->asSequence() in
  l->iterate(d:Doctor;
   acc:Sequence(Tuple(doc:Doctor,proj:Real))=Sequence{}|
    acc->append(Tuple{doc:d,
     proj:1-self.combinedFunctionalTrust(d).projection()})
    )->sortedBy(proj)->collect(doc)
```

Listing 1 Sorting doctors according to trust.

We can see that it iterates over the list of doctors of the given speciality and computes, for each one, a real number that corresponds to 1.0 minus the projection of the subjective opinion that represents the trust that the DA has in that doctor. Then, we sort the collection according to these values and return the doctors only, in that order. We have to subtract the projection from 1.0 because the OCL operation sortedBy sorts the elements of the collection in ascending order of value, and we want the doctors with the highest trust first. Operation combinedFunctionalTrust(trustee) is in charge of computing the trust of the DA user on the trustee.

3.3. Representing Reputation

As for trust, we will use opinions in Subjective logic to represent reputation. In this way we will be able to express degrees of uncertainty about the reputation of an agent, and also to combine reputation with trust using Subjective logic standard operators. Thus, in our proposal the reputation of a given agent will be stored as a pair (rep, exp) where rep is a Subjective opinion and exp represents the expiration date of that opinion.

To incorporate reputation into our high-level architecture, Fig. 2 adds a new class to Fig. 1, ReputationOpinion (colored in pink). Thus, reputation is stored as a DA record (it inherits from class DigitalAvatarRecord), has a scope (the TrustApplication it is associated with), and an association with the reputee, who is the reputation refers to (represented by a DA record with his or her personal information). Moreover, three new operations have been added to class TrustApplication to manage reputation.

The first one, reputation(person), allows the DA of a user to consult the reputation of another person by looking it up in its contacts. This method is specified in OCL in Listing 2.

```
(self.reputation->select(reputee=person)
  ->select(ro| ro.expirationDate<=self.da.clock.now)
  ->any(true)).reputation
```

Listing 2 Obtaining the reputation of a trustee.

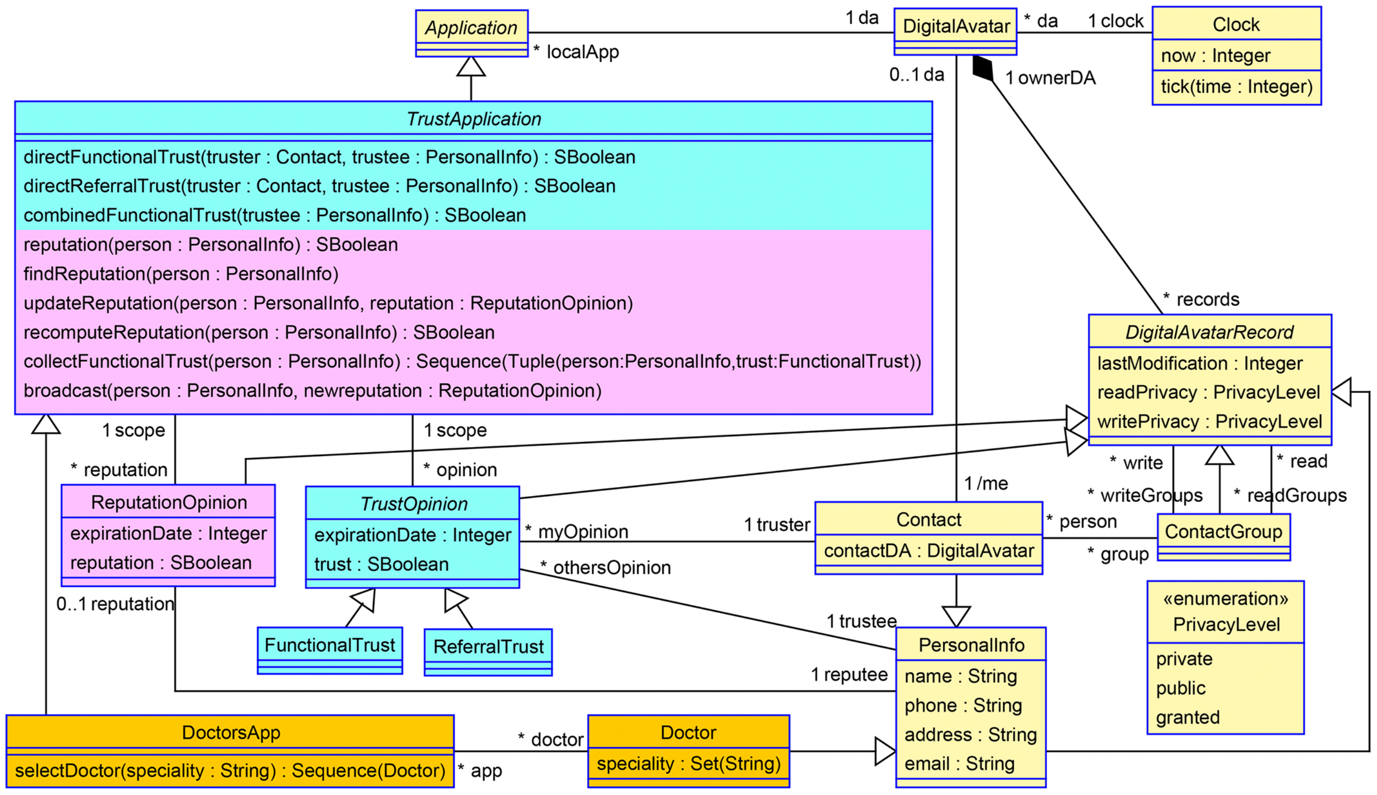
We can see that it simply looks for the ReputationOpinion record whose associated reputee coincides with the corresponding person, and returns the reputation stored in that record. If no such record exists, or the record has expired, this operation returns Undefined, i.e., null.

When this operation returns null, the DA can ask its contacts to check if any of them knows the reputation of that person. Operation findReputation(person) is in charge of that task. Its specification in SOIL is shown in Listing 3.

```
findReputation(person:PersonalInfo)
-- finds a person's reputation by asking its DA contacts
begin
  declare contacts: Sequence(Contact),
          rep:ReputationOpinion, raux:ReputationOpinion,
          found:Boolean, i:Integer;
  -- select those contacts who also have the DoctorsApp
  contacts:=self.da.records->select(oclIsKindOf(Contact))
    ->excluding(self.da.me)->select(c|c.ownerDA.localApp->
      select(a|a.oclIsKindOf(DoctorsApp))
    ->notEmpty())->oclAsType(Set(Contact))->asSequence();
   -- We iterate over those contacts until we find one
   -- with the reputation of the person we are looking for
   found:=false; i:=1;
   while (not found) and (i <=contacts->size()) do
     if ((contacts->at(i)).ownerDA.localApp->
        select(a|a.oclIsKindOf(DoctorsApp))->
         any(true)).oclAsType(TrustApplication).
         reputation(person)<>null then
        raux := ((contacts->at(i)).ownerDA.localApp->
         select(a|a.oclIsKindOf(DoctorsApp))->
         any(true)).oclAsType(TrustApplication).
         reputation->select(r|r.reputee=person)->any(true);
        rep := new ReputationOpinion();
        rep.reputation:=raux.reputation;
        rep.expirationDate:=raux.expirationDate;
        insert(rep,person) into ReputationPerson;
        insert(rep,self) into ReputationScope;
        found:=true;          -- No need to continue
      end;
      i:=i+1;
    end;
end
```

Listing 3 Looking for a reputation by asking the DA contacts.

Roughly speaking, it asks its DA contacts who use the same application until one of them knows the reputation of the person we are looking for. If found, it creates a ReputationOpinion



Application
* localApp
1 da
DigitalAvatar
* da
1 clock
Clock
now : Integer
tick(time : Integer)
0..1 da
1 ownerDA
TrustApplication
directFunctionalTrust(truster : Contact, trustee : PersonalInfo) : SBoolean
directReferralTrust(truster : Contact, trustee : PersonalInfo) : SBoolean
combinedFunctionalTrust(trustee : PersonalInfo) : SBoolean
reputation(person : PersonalInfo) : SBoolean
findReputation(person : PersonalInfo)
updateReputation(person : PersonalInfo, reputation : ReputationOpinion)
recomputeReputation(person : PersonalInfo) : SBoolean
collectFunctionalTrust(person : PersonalInfo) : Sequence(Tuple(person:PersonalInfo,trust:FunctionalTrust))
broadcast(person : PersonalInfo, newreputation : ReputationOpinion)
* records
DigitalAvatarRecord
lastModification : Integer
readPrivacy : PrivacyLevel
writePrivacy : PrivacyLevel
1 scope
1 scope
* reputation
* opinion
* write
* read
* writeGroups
* readGroups
1 /me
ReputationOpinion
expirationDate : Integer
reputation : SBoolean
TrustOpinion
expirationDate : Integer
trust : SBoolean
* myOpinion
1 truster
Contact
contactDA : DigitalAvatar
* person
ContactGroup
* group
* othersOpinion
«enumeration»
PrivacyLevel
private
public
granted
0..1 reputation
FunctionalTrust
ReferralTrust
1 trustee
PersonalInfo
name : String
phone : String
address : String
email : String
1 reputee
DoctorsApp
selectDoctor(speciality : String) : Sequence(Doctor)
* doctor
* app
Doctor
speciality : Set(String)

Figure 2 Class Diagram of the DA framework incorporating reputation (shaded in pink).

record for that person; otherwise nothing is done. Note that this works because the reputation of a person in a community is the same for all nodes. A different problem is how to compute that reputation in a P2P environment, which is discussed next.

4. Computing Reputation

To calculate the reputation of an agent in the context of Social Computing applications, we need to address two different problems. Since reputation is defined as an aggregation of the individual opinions (trusts) of the members of a community in the context of an application, we therefore need to first (*a*) collect these individual opinions, and then (*b*) aggregate them in some way. The next two sections deal with these two issues, in reverse order. We shall first see in Sect. 4.1 how the individual opinions, once collected, can be merged. Then, Sect. 4.2 describes how to collect the opinions of the members of a community to calculate the reputation of a person.

4.1. Merging individual opinions

Computing reputation means aggregating, in some way, the individual opinions that users within a community have about a person, to derive a joint opinion. Assuming that we have collected them (see next Sect. 4.2), Subjective logic provides with the perfect set of fusion operators to merge opinions (Jøsang 2016). This is an important advantage of using Subjective logic to represent the users' individual opinions, i.e., their trusts.

Of these fusion operators, which were introduced in Sect. 3.1, the one that best calculates reputation is the *Epistemic Cumulative Belief Fusion* (ECBF), for two main reasons. First, it assumes independent opinions, as is our case. Second, it is epistemic (and not aleatory) since the uncertainty involved is due to lack of knowledge, and not statistical in nature. The way this fusion operator deals with conflicting opinions also suits our needs. In case of divergent opinion, the uncertainty of the resulting opinion grows, adequately balancing the opinions.

4.2. Collecting individual opinions

The process to collect the individual opinions on a given person is highly dependent on the global architecture of the system: peer-to-peer or centralized. Although our appproach is clearly peer-to-peer, in this section we will describe how to proceed with the trust collection process in both settings.

4.2.1. Collecting opinions in server-centric settings.

Today, a fairly high percentage of social computing is still developed under a server-centric approach. Applications from Facebook, Google, Twitter or Amazon follow this approach by providing services to their users in exchange for reserving the rights to use the data shared/published by them, which is stored on cloud servers. This raises serious privacy and content ownership issues and requires placing a great deal of trust in the service providers. This is why our proposal delegates the ownership of this data exclusively to the user, who shares it in a secure manner with third parties.

Under the server-centric paradigm and keeping the ownership of the data in the user, our framework allows to calculate the

```
computeReputation(person:PersonalInfo)
begin
  declare combinedFT:Sequence(SBoolean),
          ro:ReputationOpinion;
  -- first, we collect all trust opinions
  ro:=new ReputationOpinion;
  combinedFT := self.clients->iterate(c;
    s:Sequence(SBoolean) = Sequence{} | let o:SBoolean=
        c.directFunctionalTrust(c.da.me,person) in
        if o=null then s else s->append(o) endif );
  if combinedFT->notEmpty() then -- at least one trust
     -- we merge them using the ECBF fusion operator
    ro.reputation:= let f:SBoolean=combinedFT->first() in
      f.epistemicCumulativeBeliefFusion(combinedFT->
                                         excluding(f));
    ro.expirationDate:=self.clock.now+self.expirationTime;
    -- finally, we update all clients with the new info.
    for l in self.clients do
      l.updateReputation(person,ro);
    end
  end
end
```

Listing 4 Assigning reputation to a person in a centralized environment.

reputation in a centralized way. For this, we suppose that there is a centralized application (`AppManager`) that is known to the Apps running in the Digital Avatar of a user (which are of type `TrustApplication`).

Listing 4 shows in detail the OCL specification of the method that computes the reputation of a person in a centralized setting, in the context of an application. We assume that this operation is executed by the `AppManager` central application, and `self.clients` is the set of local applications running in the DAs of the app users. It first collects the direct trust opinions of all its clients about the person using their `directFunctionalTrust()` methods. Then, it aggregates them using the ECBF fusion operator to compute the global reputation of the person. Finally, it updates the information about the reputation of the person using the clients' `updateReputation()` method.

This method uses attribute `ExpirationTime` of class `AppManager` that defines how long it takes for a reputation to expire, and which is added to the current time to calculate the expiration date to include in the `ReputationOpinion` record.

4.2.2. Collecting opinions in peer-to-peer settings. Our proposal follows the People-as-a-Service (PeaaS) model where each user is a service provider managing its own information, and interacting with the rest of the users by exchanging messages with them. Under this approach, reputation can be calculated in a distributed way by making use of the fact that the DAs of the users constitute a network of nodes. We assume that the communication subsystem consists of reliable point-to-point channels, i.e., the protocol for sending point-to-point messages between DAs is reliable. We also assume that the distributed system is, in principle, asynchronous, i.e. there are no known time limits on message latency, message processing times, or the duration of operations. This is precisely the environment of any application in the Digital Avatars framework.

Under these assumptions, there are several circumstances in which a user may require his local `TrustApplication` to update the reputation of a subject by invoking the `recomputeReputation()` method: either because new users have joined the network whose opinions have not yet been taken into account, or because the current reputation value in the system has expired. In any of these cases, the `recomputeReputation()` method distinguishes three distinct phases:

- Phase 1: Choice of the leader that is going to manage the computation of the reputation.
- Phase 2: The computation of the new reputation value.
- Phase 3: Dissemination (broadcast) of the newly calculated value throughout the network.

Phase 1. Choosing a leader. In the distributed algorithms literature, the distributed consensus problem is a well-studied problem for which different solution exists. In our context, the associativity of the Epistemic Cumulative Belief Fusion (ECBF) operator that we use for computing the reputation from the trusts of the individual users depends on the preservation of relative weights of intermediate results (Jøsang 2016), which means that no local solutions can be computed to be later aggregated to independently calculate the final value. This implies that we will need a leader node that gathers all values, computes the reputation, and disseminate that value throughout the network.

The PAXOS algorithm proposed by Lamport (Lamport 2006) and recently, the RAFT algorithm (Fazlali et al. 2019), are algorithms based on the election of a leader that manages the process of computing the consensus value. Our proposal adopts the RAFT algorithm for the election of that leader node.

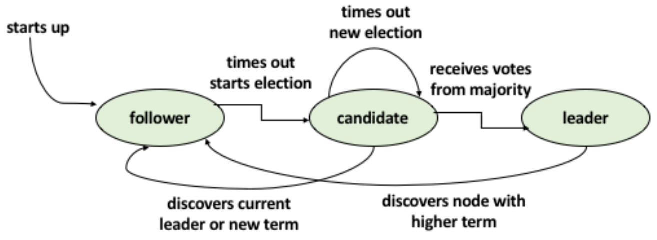


Figure 3 State diagram of the RAFT Algorithm.

As shown in Figure 3, every local `TrustApplication` can be running under three states: follower, candidate or leader state. Initially all `TrustApplications` start in the follower state. In the absence of a leader in the network, any `TrustApplication` can request to be the leader that manages the reputation computation. This request (`runningForLeader()`) is made by a node to its neighboring nodes and has associated with it the identifier of the node to be established as leader (`contactDA`) and a timestamp. When a node receives this message, if its identifier does not match that of the request, it broadcasts the received message so that the message is propagated throughout the network. From the time a node initiates a `runningForLeader()` until it enters the candidate state, it waits for a timeout called election timeout (time required for the message to be broadcast over the network).

After the election timeout, the follower becomes a candidate and starts a new election term. It sends out a `requestVote()` messages to the other nodes. If the receiving node hasn't voted yet in this term then it votes for the candidate (`vote()`) and the node resets its election timeout. When the candidate node receives a sufficient number of votes (half the number of nodes plus one) it becomes the leader, as shown in Fig. 3.

Phase 2. Computing the reputation. As mentioned above, the opinions need to be merged using the ECBF operator in one node, since the associativity of this operator depends on the preservation of the relative weights of the intermediate results (Jøsang 2016). This has an important impact on the computation of reputation, since the intermediate nodes of the network (those in the follower state) will not be able to perform partial computations of reputation as it happens with other distributed problems such as the dynamic average consensus (Mehyar et al. 2005; Kia et al. 2018). Instead, follower nodes should simply propagate and collect the functional trust opinion from their neighboring nodes and build with that data a sequence of `FunctionalTrust` opinions to deliver to their parent nodes.

This is exactly the purpose of method `collectFunctionalTrust()`. When a node receives such a message, it queries all the contacts in its DA that share the same application. It forwards the `collectFunctionalTrust()` message to each of these nodes so that the collection of information is spread throughout the network. The response to that message, from a node, is a `Sequence` of pairs `(person,functionalTrust)` initialised with the `FunctionalTrust` opinion of the node receiving the request and its identification. To this `Sequence`, the node receiving the request appends the `FunctionalTrust` opinions of its contacts as well as their respective identifiers. Including in this sequence the identifiers of the consulted DAs aims to avoid duplicity in the collected information since the network may contain cycles, which could cause duplicated records.

Finally, the node leading the reputation update process goes through these sequences of tuples, excludes possible duplicates and applies the ECBF operator on the `FunctionalTrust` opinions collected. This process is graphically described in the sequence diagram shown in Fig. 4.

Phase 3. Updating the reputation in all nodes. Once the leader has computed the reputation, this phase consists only of updating the records stored in the DAs with the new value. The leader will send a message to its direct contacts (`broadcast(person,newReputation)`) for them to broadcast the update. When a node receives that message, it stores the new reputation in its DA and propagates the message to its neighboring nodes.

Periodically, the DA runs a process that deletes `TrustOpinion` and `ReputationOpinion` records whose validity has expired. This guarantees the DA only contains those opinions that we can consider suitable and valid to carry out computations with them.

Trust	Reputation	WBF
(0.0, 0.0, 1.0, 0.5)	(1.0, 0.0, 0.0, 0.5)	(1.0, 0.0, 0.0, 0.5)
(0.0, 0.0, 1.0, 0.5)	(0.0, 1.0, 0.0, 0.5)	(0.0, 1.0, 0.0, 0.5)
(1.0, 0.0, 0.0, 0.5)	(0.0, 0.0, 1.0, 0.5)	(1.0, 0.0, 0.0, 0.5)
(0.0, 1.0, 0.0, 0.5)	(0.0, 0.0, 1.0, 0.5)	(0.0, 1.0, 0.0, 0.5)
(0.9, 0.0, 0.1, 0.5)	(0.7, 0.0, 0.3, 0.5)	(0.86, 0.0, 0.14, 0.5)
(0.0, 0.9, 0.1, 0.5)	(0.0, 0.7, 0.3, 0.5)	(0.0, 0.86, 0.14, 0.5)
(0.7, 0.0, 0.3, 0.5)	(0.9, 0.0, 0.1, 0.5)	(0.86, 0.0, 0.14, 0.5)
(0.0, 0.7, 0.3, 0.5)	(0.0, 0.9, 0.1, 0.5)	(0.0, 0.86, 0.14, 0.5)
(0.9, 0.0, 0.1, 0.5)	(0.0, 0.9, 0.1, 0.5)	(0.45, 0.45, 0.1, 0.5)
(0.0, 0.9, 0.1, 0.5)	(0.9, 0.0, 0.1, 0.5)	(0.45, 0.45, 0.1, 0.5)

Table 1 Combining Trust and Reputation with the WBF operator.

5. Marrying Reputation with Trust

Returning to the example presented in Sect. 2.4, Alberto needs to decide which cardiologist he should call. On the one hand, the app has allowed him to calculate the reputation of the different specialists and to select, in principle, the one with the best reputation. To gain more knowledge about this candidate, he has consulted his direct contacts about their trust in that doctor, and has formed an opinion by unifying these opinions using the `combinedFunctionalTrust()` operator. At this point, Alberto relies on two pieces of information to make a decision: the overall reputation of the specialist in question, and the confidence in him derived from the experiences of his direct contacts.

To merge reputation and trust, the Weighted Belief Fusion (WBF) fusion operator from Subjective logic seems to be the best option. To better understand how this operator works, Table 1 illustrates with examples how different opinions are merged. We can see how the more certain opinions (i.e., those with less uncertainty) have a greater weight in the resulting opinion and, therefore, prevail. In the case of discrepant opinions with low uncertainty, in the resulting opinion the degrees of belief and disbelief adopt intermediate values keeping uncertainty low. In this case there is no clear winning opinion.

In real scenarios, personal experience or direct information by our contacts usually has more weight than the global reputation calculated by the system. It is only in the absence of knowledge from our contacts, or when the degree of uncertainty in our contacts' opinions is high, when our opinion is based on reputation. Therefore, when the fusion operator does not provide a clear winning decision, we normally use trust. In Subjective logic terms, when the projection of the result is close to 0.5, the combined trust obtained from our contacts' opinion should prevail.

Listing 5 shows how the `selectDoctor()` operation works now, using the WBF fusion operator to combine trust and reputation when sorting the doctors.

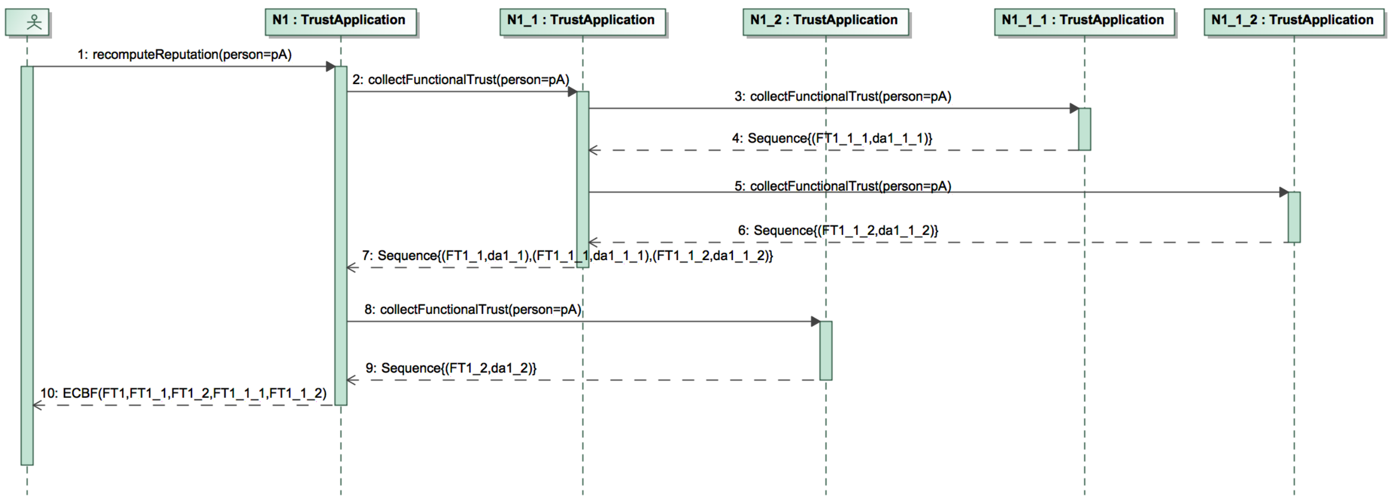

N1 : TrustApplication
N1_1 : TrustApplication
N1_2 : TrustApplication
N1_1_1 : TrustApplication
N1_1_2 : TrustApplication
1: recomputeReputation(person=pA)
2: collectFunctionalTrust(person=pA)
3: collectFunctionalTrust(person=pA)
4: Sequence{(FT1_1_1,da1_1_1)}
5: collectFunctionalTrust(person=pA)
6: Sequence{(FT1_1_2,da1_1_2)}
7: Sequence{(FT1_1,da1_1),(FT1_1_1,da1_1_1),(FT1_1_2,da1_1_2)}
8: collectFunctionalTrust(person=pA)
9: Sequence{(FT1_2,da1_2)}
10: ECBF(FT1,FT1_1,FT1_2,FT1_1_1,FT1_1_2)

Figure 4 Sequence Diagram of Phase 2. The leader node sends a message to its followers which simply propagate and collect the functional trust opinion from their neighboring nodes. Finally, the leader applies the ECBF operator on the collected opinions.

6. Implementation

This section describes how the `DoctorsApp` application is implemented in Android using the Digital Avatars Framework.

To include all the functionality related with trust management, every DA needs to store and manage a set of trust records. The `TrustOpinion` records are used to specify the phone owners' functional and referral trust in their contacts. These records are scope dependent and therefore they need to be associated to a specific application — in this case, the `DoctorsApp` application. Another set of `TrustOpinion` records is needed, which represent the trust in the doctors of each of the contacts for whom the user has a referral trust. There are two different ways of obtaining these records. When the user wants to know her functional trust in a doctor and such a record does not exist, the DA sends a message to all the contacts which are trustees of a referral opinion of that user, asking them for their functional trust on the doctor. Once the replies are received, the user's functional trust is computed using the ECBF operator. Alternatively, we can use a background process that is in charge of maintaining the trust records of the DA contacts, collecting them from the DA's contacts when they are not available or have expired. This way, the application can assume that all the information needed for computing the trust of a user is available and updated. This is the approach chosen to implement the Android application as it allows more efficient trust management and calculation.

A similar process applies to reputation management. As reputation is a global property, each user will always have the same reputation value for a given doctor. Such a value is calculated by adequately combining the functional trust of all users of the application (see Sect. 4). Therefore, a background process approach is also suitable for reputation management, which is responsible for updating the `ReputationOpinion` records when they expire or when a new user installs the `DoctorsApp` in their DA.

With the trust and reputation information available and updated, the `selectDoctor()` operation is easily implementable as specified in Listing 5 below. To list the specialists in order of confidence, the algorithm considers the direct functional trust of the user and the reputation of the doctor. The application uses the Weighted Cumulative Fusion operator to combine these two `SBoolean` values, and uses the projection as sorting key.

```
selectDoctor(speciality:String) : Sequence(Doctor) =
  let l:Sequence(Doctor) = self.doctor->
          select(d|d.speciality->includes(speciality))
           ->asSequence() in
  l->iterate(d:Doctor;
      acc:Sequence(Tuple(doc:Doctor,proj:Real))=Sequence{} |
         acc->append(Tuple{doc:d,
            proj:let r:SBoolean = self.reputation(d) in
                 let t:SBoolean =
                        self.combinedFunctionalTrust(d) in
                 1.0 - r.weightedBeliefFusion(Sequence{t}).
                          projection()
                })
    )->sortedBy(proj)->collect(doc)
```

Listing 5 Sorting doctors according to trust and reputation combined.

The complete code of the example application, along with the current version of the Digital Avatars framework, can be found on GitHub (Pérez-Vereda 2022). The implementation uses the Java library with the datatypes extended with uncertainty and in particular datatype `SBoolean`, which represents trust opinions and its operations (Atenea Research Group 2021). DA records are stored as JSON documents using a CouchBase Lite NoSQL database.[1] Messages among the DA applications are managed and transmitted as events using the Complex Event Processing (CEP) engine of the framework. It has been implemented using Siddhi's lightweight CEP engine (Suhothayan et al. 2011) 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, in addition to the typical features of any CEP engine of receiving, handling events and sending events.

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

7. Related Work

Systems based on the reputation computation have attracted the attention of the scientific community for some years now. The literature gathers numerous research works in different areas that affect distributed environments, from peer-to-peer (P2P) applications, ad-hoc proposals, and systems based on social networks. These proposals have in common the existence of entities that offer services to third parties, whose goal is help deciding the best option to choose among a set of service providers. In contrast, the existing proposals differ in other aspects such as the scope of the reputation (whether it is global or local, depending on whether the calculated value is based on global information or computed from personal information), and the reputation estimation algorithm used.

7.1. Reputation computing based on subjective logic

Subjective logic allows to represent and operate with opinions that have some degree of uncertainty. Subjective logic has been used by some authors to model trust networks and reputation systems. In (Santini 2019), entities can develop their opinion taking into account their direct trust and the beliefs of their contacts through the trust pathways that link them. All these subjective opinions can be merged into a reputation score calculated using the Consensus & Compromise Fusion (CCF) operator. This score represents the degree of credibility that the community of agents assigns to the entity. Unlike our proposal, this paper calculates a local reputation and uses a fusion operator that neutralizes conflicting opinions. From our point of view, conflicting opinions should not be neutralized but taken into the same consideration as other opinions for the computation of reputation.

Despite the advantages of Subjective logic, some authors consider that some of its operators (in particular, the discount operator) do not manage evidence in a natural way. They also point out the high dependence of Subjective logic on the structure of the network, to the point that a limit on the information collected to make the calculations should be made. To address these limitations, in (Skoric et al. 2014) the authors present a new algebra that allows calculation of reputation that is not sensitive to the structure of the network, and where no information need to be discarded. This proposal brings together two worlds: flow-based reputation systems and consistent handling of uncertainties in Subjective logic. The reputation calculated is global in this case and the algorithm used to calculate the reputation is based on the use of Markov chains. We find the proposal extremely interesting although we do not share some their criticisms about Subjective logic: (1) We do not see as a limitation that dogmatic opinions are sensitive to the consensus operator; and (2) the fact that the discount operator is not distributive with respect to the cumulative fusion operator does not contradict the intuition from our point of view either.

7.2. Reputation Evaluation in Distributed Mobile Networks

Our proposal is also closely related to systems that compute reputation in distributed mobile network environments. The nature of these mobile networks is slightly different from that of any distributed environment and therefore requires algorithms that are not only distributed but also provide results that are as competitive as some centralized proposals (Sharma et al. 2020). In (Turkina & Ihnatiev 2020) the authors propose a method to evaluate both trust and reputation in mobile networks in the context of the Internet of Things. The implemented strategy tries to determine the best object to interact with. Taking into account the results of such interaction, it adjusts previous trust estimates on the objects whose recommendations were used to evaluate trust in the selected object. This proposal differs with ours in several aspects: (a) trust and reputation values are calculated in an interval [0..1] without taking uncertainty into account; (b) the calculation of trust is performed on the basis of reputation while our proposal adopts the opposite approach. However, we have found particularly interesting the feedback they propose for modifying the previous values used in the final choice based on subsequent experiences with the selected object. We plan to consider this as part of our future work.

In (Chiejina et al. 2015), a dynamic reputation management system is proposed to detect and isolate misbehaving nodes in mobile ad hoc networks. The model uses a direct monitoring technique to evaluate the reputation of a node in the network, which ensures that nodes that expend their energy in transmitting data and routing control packets to others can carry out their activities in the network, while misbehaving nodes are detected and isolated from the network. The proposed reputation model consists of a monitor, a reputation manager, a punishment scheme and a routing manager. In particular, we are interested in the reputation manager which calculates the reputation of a node based solely on information from its direct neighbors (local computation). It is also curious in this proposal that the nodes of the network are initialized to a default reputation and that this value oscillates in the range [0..2]. After monitoring the different modes activities for a period of time, the reputation at that instant is calculated as a combination of the initial reputation and the average reputation value obtained up to that time. Unlike this proposal, our approach computes the reputation of a node from the individual direct trusts of the rest of the nodes, always from scratch because we assume that the direct trust of users is always kept updated. Moreover, our proposal does not use a centralized reputation manager, but a peer-to-peer algorithm to compute reputation.

7.3. Reputation in collaborative applications

There are numerous papers in the literature that address the calculation of reputation as a determinant variable within their collaborative system. In (Hoh et al. 2020), for example, the authors introduce the concept of collaborative on-road reputation (CORR), in which an individual driver's reputation is automatically calculated and reviewed by nearby drivers. If the reputation score goes below a threshold, drivers may experience limited kinetic vehicle movement or even temporary vehicle suspension. The proposed system assigns an initial reputation to all drivers with a maximum value, and adjusts reputation scores based on the degree of impact of anomalous driving on nearby drivers. The data collection is performed on all the vehicles comprising the cluster and it is the cloud-hosted server that is responsible for determining the source of the anomaly, its type and its level of impact, after which it updates the reputation score accordingly. The reputation thus calculated takes into account all the incidents that a user may have while driving and must be constantly evaluated, which introduces unnecessary CPU usage in the system. In our proposal, the reputation has an expiration date after which it must be recalculated and distributed.

Although our work deals with the calculation of reputation applied to individuals, it is true that reputation is a concept applicable to many other entities present in collaborative contexts. Proof of this is Wikipedia, one of the best known and most widespread collaborative applications today, where reputation is also used. The content published on Wikipedia is predominantly created by anonymous or pseudonymous authors whose knowledge and motivations are unknown. As a result, a great deal of uncertainty arises as to the quality of their contributions. One way to address this uncertainty and the problem generated by the dubious quality of content is to implement automatic reputation systems that calculate, through some kind of metric, the reputation acquired by a given wikipedia content or article. These metrics can help us determine how reliable a publication is. This idea has become a new branch of research in recent years. In (Wöhner et al. 2011), for example, seven metrics are compared, some of which come from the literature and some of which are new proposals by the authors. The metrics proposed for the calculation of reputation are validated through an analysis of different groups of Wikipedia users.

Reputation systems based on feedback management play an important role in today's online cosumer markets. These opinions, as a feedback mechanism to establish reputation, are present in well-known applications such as Google, Amazon, TripAdvisor (Buccafurri et al. 2015; Reyes-Menendez et al. 2019) or eBay (Hayne et al. 2015; Hui et al. 2016; Xie & Lui 2015) among others. Although these works present some similarities between them and with our proposal, their reputation systems diverge in several aspects, such as: whether they model both positive and negative trust (distrust), whether they model trust in the resulting trust value or decision, whether they consider the context of the trust decision, and whether they analyze the credibility of third-party recommendations. The aspect that interests us most in this comparison is the algorithmic approach taken. In (Yao et al. 2011) a comparison is made that concludes that there are five approaches for the computation of reputation in these systems:

- Counting: Reputation is calculated as the sum or average of all ratings (eBay case).
- Probabilistic: All systems in this category are based on the use of the Beta probability density function where reputation is calculated from a set of both positive and negative experiences.
- Fuzzy: Systems in this category apply fuzzy logic to express and reason about uncertainty in reputation information.
- Flow: This category includes systems that calculate reputation based on transitive trust flow.
- Other approaches.

Something that none of these approaches take into account is uncertainty. In our view, it is essential that uncertainty is considered and included in the reputation calculation. This is why we advocate the use of Subjective logic as a mechanism not only for modeling but also for computing reputation.

8. Conclusions

This contribution proposes a reputation management system for mobile-based collaborative social computing applications. It builds on the Digital Avatars framework, extending it with the explicit representation and management of reputation information about service providers, and its combination with the users' individual opinions about these service providers to make better informed decisions in peer-to-peer environments.

We have specified the proposal using high-level, platform-independent UML models, and responded to the three research questions we posed in the introduction. First, we use Subjective Logic for expressing and reasoning about reputation, and extended the DA framework for incorporating reputation and storing it in the DAs of users. Second, we have described (Sect. 4) the algorithms required to compute reputation from the users' individual opinions in a decentralised setting. Finally, we have also shown how the fusion operators provided by Subjective logic can be used to merge reputation and trust for making better informed decisions. As a proof of concept, and to demonstrate the proposal, we have developed an implementation of a mobile-based application that takes into account both reputation and trust.

This work can be continued in several directions. First, the models presented here have served us to study the practical feasibility of our proposal by analyzing the expected structure and behavior of the implementation. We have also simulated these models with the USE facilities for executing UML models (Büttner & Gogolla 2014), which has served to initially validate our proposal. We still need to explore more in depth how to ensure that the implementation conforms to the UML and OCL models. Conformance testing (Linington et al. 2011) can be of help here. Based on these models, we plan to carry out more exhaustive verification and simulation tests to analyze the system properties, employing the USE toolkit (Gogolla et al. 2018). Being able to reason about the system from its high-level models is one of the main benefits of Model-based Software Engineering.. Second, we plan to evaluate our proposal with more applications to better understand and appraise its advantages and limitations. Incorporating our trust model to other social computing application frameworks, such as (Mao et al. 2016; Bajo et al. 2016; Mohan et al. 2013; Tran et al. 2013) 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 identify trends in the evolution of the reputation of a service provider, in order to, e.g., help them modify their quality of service to improve their users' experience and thus increase their reputation.

Acknowledgments

This work is funded by the Spanish research projects PGC2018-094905-B-100 and RTI2018-098780-B-I00.

References

Adams, B., & Webb, R. (2003). *Model of trust development in small teams* (Tech. Rep. No. CR 2003-016). Department of National Defense.

Atenea Research Group. (2021). *Uncertain datatypes – Git repository.* https://github.com/atenearesearchgroup/uncertainty. GitHub.

Bajo, J., Campbell, A. T., & Zhou, X. (2016). Mobile sensing agents for social computing environments. In *Proc. of PAAMS'16* (Vol. 473, pp. 157–167). Springer. doi: 10.1007/978-3-319-40159-1_13

Bertoa, M. F., Burgueño, L., Moreno, N., & Vallecillo, A. (2020). Incorporating measurement uncertainty into OCL/UML primitive datatypes. *Softw. Syst. Model.*, *19*(5), 1163–1189. doi: 10.1007/s10270-019-00741-0

Bertoa, M. F., Moreno, N., Pérez-Vereda, A., Bandera, D., Álvarez-Palomo, J. M., & Canal, C. (2020). Digital avatars: Promoting independent living for older adults. *Wirel. Commun. Mob. Comput.*, *2020*, 8891002:1–8891002:11. doi: 10.1155/2020/8891002

Buccafurri, F., Lax, G., Nicolazzo, S., & Nocera, A. (2015). A model implementing certified reputation and its application to tripadvisor. In *Proc. of ARES'15* (pp. 218–223). IEEE Computer Society.

Burgueño, L., Bertoa, M. F., Moreno, N., & Vallecillo, A. (2018). Expressing confidence in models and in model transformation elements. In *Proc. of MODELS'18* (pp. 57–66). ACM. doi: 10.1145/3239372.3239394

Burgueño, L., Muñoz, P., Clarisó, R., Cabot, J., Gérard, S., & Vallecillo, A. (2022). Dealing with belief uncertainty in domain models. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *To appear.*.

Büttner, F., & Gogolla, M. (2014). On OCL-based imperative languages. *Sci. Comput. Program.*, *92*, 162–178. doi: 10 .1016/j.scico.2013.10.003

Chiejina, E., Xiao, H., & Christianson, B. (2015). A Dynamic Reputation Management System for Mobile Ad Hoc Networks. *Comput.*, *4*(2), 87–112. doi: 10.3390/computers4020087

de Finetti, B. (2017). *Theory of probability: A critical introductory treatment*. John Wiley & Sons. doi: 10.1002/9781119286387

de Siqueira Braga, D., Niemann, M., Hellingrath, B., & de Lima Neto, F. B. (2019). Survey on computational trust and reputation models. *ACM Comput. Surv.*, *51*(5), 101:1–101:40. doi: 10.1145/3236008

Fazlali, M., Eftekhar, S. M., Dehshibi, M. M., Malazi, H. T., & Nosrati, M. (2019). Raft Consensus Algorithm: an Effective Substitute for Paxos in High Throughput P2P-based Systems. *CoRR*, *abs/1911.01231*. http://arxiv.org/abs/1911.01231.

Feller, W. (2008). *An introduction to probability theory and its applications*. Wiley.

Gambetta, D. (2000). Can we trust trust? In *Trust: Making and breaking cooperative relations* (pp. 213–237). Univ. Oxford.

Gogolla, M., Hilken, F., & Doan, K.-H. (2018, December). Achieving model quality through model validation, verification and exploration. *Computer Languages, Systems & Structures*, *54*, 474–511. doi: 10.1016/j.cl.2017.10.001

Grandison, T., & Sloman, M. (2000). A survey of trust in internet applications. *IEEE Commun. Surv. Tutorials*, *3*(4), 2–16. doi: 10.1109/COMST.2000.5340804

Griffin, D., & Tversky, A. (1992). The weighing of evidence and the determinants of confidence. *Cognitive Psychology*, *24*(3), 411–435.

Guillén, J., Miranda, J., Berrocal, J., García-Alonso, J., Murillo, J. M., & Canal, C. (2014). People as a service: A mobile-centric model for providing collective sociological profiles. *IEEE Software*, *31*(2), 48–53. doi: 10.1109/MS.2013.140

Hayne, S. C., Wang, H., & Wang, L. (2015). Modeling reputation as a time-series: Evaluating the risk of purchase decisions on ebay. *Decis. Sci.*, *46*(6), 1077–1107.

Hoh, B., Ucar, S., Oza, P., Patnayak, C., & Oguchi, K. (2020). CORR: collaborative on-road reputation. In *Proc. of CAVS'20* (pp. 1–6). IEEE. doi: 10.1109/CAVS51000.2020.9334679

Hui, X., Saeedi, M., Shen, Z., & Sundaresan, N. (2016). Reputation and regulations: Evidence from ebay. *Manag. Sci.*, *62*(12), 3604–3616.

JCGM 100:2008. (2008). *Evaluation of measurement data – Guide to the expression of uncertainty in measurement (GUM).* http://www.bipm.org/utils/common/documents/jcgm/JCGM_100_2008_E.pdf.

Jøsang, A. (2016). *Subjective logic - A formalism for reasoning under uncertainty*. Springer.

Kia, S. S., Scoy, B. V., Cortés, J., Freeman, R. A., Lynch, K. M., & Martínez, S. (2018). Tutorial on dynamic average consensus: the problem, its applications, and the algorithms. *CoRR*, *abs/1803.04628*. http://arxiv.org/abs/1803.04628.

Lamport, L. (2006). Fast paxos. *Distributed Comput.*, *19*(2), 79–103. doi: 10.1007/s00446-006-0005-x

Linington, P. F., Milosevic, Z., Tanaka, A., & Vallecillo, A. (2011). *Building enterprise systems with odp – an introduction to open distributed processing*. Chapman & Hall/CRC Press.

Mao, H., Xiao, M., Liu, A., Li, J., & Hu, Y. (2016). OCC: opportunistic crowd computing in mobile social networks. In *Proc. of DASFAA'16* (Vol. 9645, pp. 254–267). Springer. doi: 10.1007/978-3-319-32055-7_21

McKnight, D. H., & Chervany, N. L. (2001). Conceptualizing trust: A typology and e-commerce customer relationships model. In *Proc. of HICSS-34.* doi: 10.1109/HICSS.2001 .927053

Mehyar, M., Spanos, D., Pongsajapan, J., Low, S. H., & Murray, R. M. (2005). Distributed averaging on asynchronous communication networks. In *CDC/ECC* (pp. 7446–7451). IEEE.

Miranda, J., Mäkitalo, N., García-Alonso, J., Berrocal, J., Mikkonen, T., Canal, C., & Murillo, J. M. (2015). From the internet of things to the internet of people. *IEEE Internet Computing*, *19*(2), 40–47. doi: 10.1109/MIC.2015.24

Mohan, S., Agarwal, N., & Al-Doski, L. (2013). Mobile network-aware social computing applications: a framework, architecture, and analysis. *J. Ambient Intell. Humaniz. Comput.*, *4*(1), 43–56. doi: 10.1007/s12652-011-0066-y

Muñoz, P., Burgueño, L., Ortiz, V., & Vallecillo, A. (2020). Extending OCL with Subjective Logic. *Journal of Object Technology*, *19*(3), 3:1-15. doi: 10.5381/jot.2020.19.3.a1

Muñoz, P., Pérez-Vereda, A., Moreno, N., Troya, J., & Vallecillo, A. (2021). Incorporating trust into collaborative social computing applications. In *Proc. of EDOC'21* (pp. 21–30). IEEE. doi: 10.1109/EDOC52215.2021.00020

Pérez-Vereda, A., & Canal, C. (2017). A people-oriented paradigm for smart cities. In *Proc. of ICWE'17* (Vol. 10360, pp. 584–591). Springer. doi: 10.1007/978-3-319-60131-1_46

Pérez-Vereda, A. (2022). *Digital avatars: Doctors' application – Git repository.* https://github.com/apvereda/Digital-Avatars-DoctorsApp.

Petrusic, W., & Baranski, J. (2003). Judging confidence influences decision processing in comparative judgments. *Psychonomic Bulletin & Review*, *10*, 177–183. doi: https://doi.org/10.3758/BF03196482

Reyes-Menendez, A., Saura, J. R., & Martinez-Navalon, J. G. (2019). The impact of e-wom on hotels management reputation: Exploring tripadvisor review credibility with the ELM model. *IEEE Access*, *7*, 68868–68877.

Richters, M., & Gogolla, M. (1998). On formalizing the UML object constraint language OCL. In *ER* (Vol. 1507, pp. 449–464). Springer.

Richters, M., & Gogolla, M. (2000). Validating UML models and OCL constraints. In *UML* (Vol. 1939, pp. 265–277). Springer.

Santini, F. (2019). From trust among agents to reputation of abstract arguments by using subjective logic. In *Proc. of AI*IA'19* (Vol. 2528, pp. 65–79). CEUR-WS.org. http://ceur-ws.org/Vol-2528/6_Santini_AI3_2019.pdf.

Schuler, D. (1994). Social computing - introduction to the special section. *Commun. ACM*, *37*(1), 28–29. doi: 10.1145/175222.175223

Sharma, V., You, I., Andersson, K., Palmieri, F., Rehmani, M. H., & Lim, J. (2020). Security, Privacy and Trust for Smart Mobile- Internet of Things (M-IoT): A Survey. *IEEE Access*, *8*, 167123–167163. doi: 10.1109/ACCESS.2020.3022661

Skoric, B., de Hoogh, S., & Zannone, N. (2014). Flow-based reputation with uncertainty: Evidence-based subjective logic. *CoRR*, *abs/1402.3319*. http://arxiv.org/abs/1402.3319.

Suhothayan, S., Gajasinghe, K., Narangoda, I. L., Chaturanga, S., Perera, S., & Nanayakkara, V. (2011). Siddhi: a second look at complex event processing architectures. In *Proc. of SC@GCE'11* (pp. 43–50). ACM. doi: 10.1145/2110486.2110493

Tran, H. M., Huynh, K. V., Vo, K. D., & Le, S. T. (2013). Mobile peer-to-peer approach for social computing services in distributed environment. In *Proc. of soict'13* (pp. 227–233). ACM. doi: 10.1145/2542050.2542064

Troya, J., Moreno, N., Bertoa, M. F., & Vallecillo, A. (2021). Uncertainty representation in software models: A survey. *Software and Systems Modeling*, *20*(4), 1183—1213. doi: 10.1007/s10270-020-00842-1

Turkina, V., & Ihnatiev, D. (2020). Approach to sustainable trust and reputation evaluation in distributed mobile networks of the internet of things. In *Proc. of DESSERT'20* (pp. 117–121). IEEE. doi: 10.1109/DESSERT50317.2020.9125015

Wöhner, T., Köhler, S., & Peters, R. (2011). Automatic reputation assessment in wikipedia. In *Proc. of ICIS'11.* Association for Information Systems. http://aisel.aisnet.org/icis2011/proceedings/onlinecommunity/5.

Xie, H., & Lui, J. C. S. (2015). Modeling ebay-like reputation systems: Analysis, characterization and insurance mechanism design. *Perform. Evaluation*, *91*, 132–149.

Yao, W., Chu, C., & Li, Z. (2011). Leveraging complex event processing for smart hospitals using RFID. *Journal of Network and Computer Applications*, *34*(3), 799–810.

About the authors

Nathalie Moreno is Lecturer at the University of Málaga. Her research interests include Model-based Engineering, Uncertainty modeling and propagation, and Trust & Reputation Systems. You can contact the author at nmv@uma.es.

Alejandro Pérez-Vereda is a PhD candidate at the University of Castilla-La Mancha. His research interests include Mobile Computing, Context-Awareness, Pervasive Systems and CrowdSensing. You can contact the author at alejandro.pvereda@uclm.es.

Antonio Vallecillo is full Professor at the University of Málaga, where he leads the Atenea Research Group on Software and Systems Modeling. His main research interests include Open Distributed Processing, Model-based Engineering and Software Quality. You can contact the author at av@uma.es or visit http://www.lcc.uma.es/~av/.