An Auditable Reputation Service for Collective Adaptive Systems

Heather S. Packer, Laura Drăgan and Luc Moreau

1 Introduction

A subject's reputation is a measure of how much a community rates it. Reputation plays a core role in online communities such as eBay and StackExchange since it can influence the communities perception and interactions, and affect computational activities within the system. In eBay, the reputation of a seller can help a buyer decide whether they want to purchase an item from this seller. In StackExchange, reputation is a key incentive for people to contribute as it leads to kudos and potential employment offers.

A subject's reputation is computed from feedback about it, which may be of two kinds:

1. User feedback consists of ratings or comments provided by users who participate in the system, and have interacted with the subject.
2. System feedback consists of various metrics directly measurable by the system, including performance, timeliness and responsiveness.

Reputation can be evaluated either manually or automatically, on a set of criteria which differs across domains.

For a subject to achieve and maintain a good reputation, it is important to understand how different factors influence its calculation. Given that reputation varies over time as feedback is submitted, it is desirable for a reputation provider to be accountable. In order to be accountable it is required to explain how reputation measures have been computed over time, which feedback reports were considered, and

Heather S. Packer
University of Southampton, e-mail: hp3@ecs.soton.ac.uk

Laura Drăgan
University of Southampton, e-mail: lcd@ecs.soton.ac.uk

Luc Moreau
University of Southampton, e-mail: l.moreau@ecs.soton.ac.uk

how the measure evolved. Hence, auditing is a key mechanism that a reputation provider can offer its clients to inspect reputation measures it derives. In this context, provenance can be used to provide descriptions of the entities, activities, and agents that may have influenced a reputation measure.

While the use of reputation is frequent in Collective Adaptive Systems (CAS), there is a lack of agreed methods for its use, representation, and auditability. The aim of this chapter is to investigate key facets of an auditable reputation service for collective adaptive systems, summarised by the following contributions:

1. Use cases for reputation and provenance in collective adaptive systems, which are categorised into functional, auditable, privacy and security, and administrative.
2. A RESTful Reputation API, which allows users access to subject feedback and to access feedback reports and reputation measures.
3. An upper level ontology for describing entities used in social machines, which is used to classify objects in provenance.

In Section 2 we outline related work on trust and reputation, and social computation. Following that, in Section 3, we describe generic provenance use cases. Then in Section 3, we discuss a reputation API and an upper level ontology for provenance and social computations (see Sections 4.1 and 4.2, respectively). In Section 5, we describe in detail a use case for a ride share application. Finally, Section 6 concludes the paper.

2 Background and Related Work

The following sections define provenance, trust and reputation, and describe their use in the context of social machines.

2.1 Provenance

Provenance has varied emerging applications: it may be used to make social computations accountable and transparent [1]; provenance can help determine whether data or users can be trusted [2]; and provenance can be used to ensure reproducibility [3] of computations.

In this chapter, we adopt the W3C (World Wide Web Consortium) Provenance Working Group's definition, given in the PROV Data Model specification. [4]:

> Provenance is defined as a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of data or a thing.

PROV is a recent recommendation of the W3C for representing provenance on the web. PROV is a conceptual data model (PROV-DM [4]), which can be mapped and

serialized to different technologies. There is an OWL2 ontology for PROV (PROV-O [5]), allowing mapping to RDF, an XML schema for provenance [6], and a textual representation for PROV (PROV-N [7]). In section 4.1 we show an extension of the PROV ontology, in the description of the Provenance for Social Computations ontology.

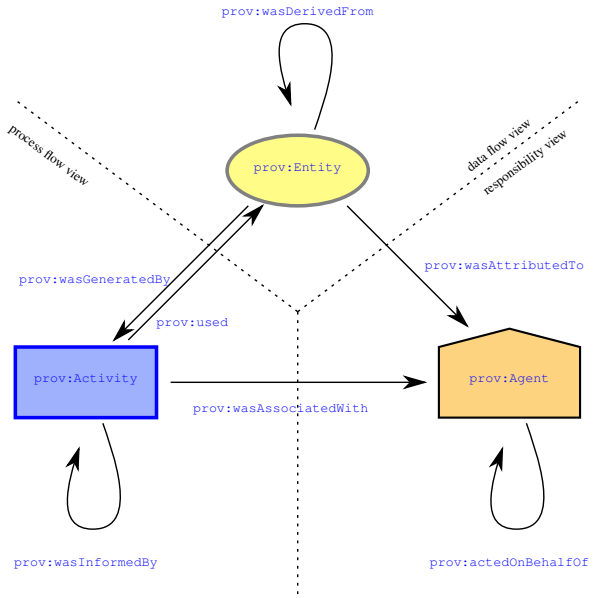


Fig. 1: Three Different Views of the Core of PROV. The figure adopts the PROV layout conventions: an entity is represented by a yellow ellipsis, an activity by a blue rectangle, and an agent by an orange pentagon. We note here that the diagram is a "class diagram" illustrating the classes that occur as domain and range of properties. Taken from [8].

2.2 Trust and Reputation

The topic of trust and reputation has been extensively reviewed ([9, 10, 11, 12]), with Artz and Gil's [11] being one of the more comprehensive survey of definitions and existing systems. Sabater and Siera [10] propose a set of criteria for the characterisation of computational models, and then proceed to qualify a selection of existing models according to these criteria. Pinyol and Sabater-Mir [13] more recently surveyed trust and reputation models for open multi-agent systems, a category to which social machines belong. This second survey cites the previous characterisation criteria as just one of several existing classification systems.

Goldbeck's work, which spans several years [14, 15, 16, 12, 17] tackles the way trust and reputation is defined and computed in online social networks. Golbeck describes trust and reputation as "socially loaded" terms, and reputation as a "more social notion of trust" [14]. The general definition of trust used in her PhD thesis [15] is:

> Alice trusts Bob if she commits to an action based on a belief that Bob's future actions will lead to a good outcome.

Different types of trust are described and studied in the research literature, and trust is often equated with the mechanism for authentication, such as digital signatures and encryption. The survey by Artz and Gil [11] lists the diverse research areas of trust, from security and access control to decision making and game theory, differentiating between policy-based and reputation-based trust – the former focusing on "hard security" and the latter on "social" aspects. The survey focuses on trust representations in the Semantic Web.

In the context of the Semantic Web, Tim Berners-Lee envisioned [18] that provenance is a crucial component in establishing trust. In [19], Berners-Lee introduces the idea of an easy way to access provenance information provided on websites with an "oh yeah?" button:

> At the toolbar (menu, whatever) associated with a document there is a button marked "Oh, yeah?". You press it when you loses that feeling of trust. It says to the Web, "so how do I know I can trust this information?".

The idea is that if we can determine where data and documents come from, we can decide whether it can be trusted. Li et al [20] outline how trust can be developed, or distrust minimised, through provenance on the Semantic Web, by describing and generalizing several use cases where possible "distrust events" occured. Prat and Madnick [21] define a provenance-grounded measure of the believability of data on the web, relying on a measure of trustworthiness of an agent as one of three dimensions.

A large amount of research in the area of trust comes from the multi-agent domain [9, 22, 13]. In a multi-agent context [9], trust is defined as:

> Trust is a belief an agent has that the other party will do what it says it will (being honest and reliable) or reciprocate (being reciprocative for the common good of both), given an opportunity to defect to get higher payoffs.

The focus of multi-agent systems trust is on actions performed by agents. It is to be distinguished from trust of information (or content trust [23]) defined as follows:

> Content trust is a trust judgement on a particular piece of information in a given context.

In this chapter we use a definition of trust that is based on a mix of provenance and reputation and content-based trust, not including the hard security.

Trust and reputation models typically introduce a measure, to represent how trustworthy or reputable a system or individual is. This measure is then typically summarised in a value, discrete (e.g. 1 to 9 [16], or 1 to 5 star rating as used by

online commerce sites like Amazon and Ebay) or continuous (e.g. [0,1] [24]), or a label (trustworthy/not trustworthy). This value is then used to make trust decisions.

Reputation about a subject is acquired from the experiences of others in a community. According to [14] "reputation is synonymous with the measure of trust", and from a more social perspective [11]:

> Reputation is an assessment based on the history of interactions with or observations of an entity, either directly via personal experience or as reported by others.

Reputation systems can be centralized or distributed[25, 26, 27, 28]. Liu [29] describes criteria for classifying and analysing centralised reputation systems. Social machines which are centralised usually use a centralised reputation system. However, some aspects in the reputation system can be decentralised, where participants can set preferences which favour or reject some sources, without taking into consideration the reputation, thus trusting or distrusting them implicitly – an example is TrustMail described by Golbeck and Hendler [14].

2.3 Provenance, Trust and Reputation in Collective Adaptive Systems

Berners-Lee and Fischetti [18] define a social machine, which are a synonym for CAS, as systems "in which the people do the creative work and the machine does the administration", where both human and machines contribute to the completion of a task which they could not achieve separately. The characterisation of social machines is also described in the Chapter [30][1] entitled "A Taxonomic Framework for Social Machines".

A number of terms and research areas involve the intersection of social behaviour and computational systems: social computing, crowdsourcing, collective intelligence, human computation [31]. Social machines are used for a large variety of tasks, too complex for humans or computers to achieve independently, but which can be divided in small simpler tasks for one of the other. These include annotation, disaster relief, mapping, collaborative filtering, online auctions, prediction markets, reputation systems, computational social choice, tagging, and verification. Many existing systems employ strategies which can qualify them as social machines, including Wikipedia, OpenStreetMap, Ushahidi, re-Captcha.

The environment for collaboration in social machines varies from system to system, it can be loosely mediated as in Twitter, or under stricter control of community policies and guidelines like in Wikipedia. Stein and Hess [32] show that in the German Wikipedia the quality of contributions is connected to the reputation of participants. The ability to uniquely identity and assess participants inputs in a collective based on their past actions or perceived domain knowledge in the system, is a factor

[1] Daniele fix this reference please

in measuring the trust and reputation of the collective output[2]. Although some social machines go out of their way to prevent uniquely identifying users, such as 4chan, most will have a way of identifying participants, usually through user accounts.

The adoption of a social machine depends on a combination of many factors, some of them include:

- the purpose set out when the social machine was created,
- the perceived benefits to the participants,
- the amount and type of tasks, or
- the level of participation required.

Human participants have to trust that the machine will deliver the expected outcomes and that any information provided by them will be used for the purpose which was described. Specifically, the users must trust that the machine will not do anything with the information that they provide, which conflicts with the purpose for which this data was captured.

Weitzner et al. [1] argue that, for information, "accountability must become a primary means through which society addresses appropriate use". For them, "information accountability means the use of information should be transparent, so it is possible to determine whether a particular use is appropriate under a given set of rules, and that the system enables individuals and institutions to be held accountable for misuse". Dynamically assembled systems need to be made accountable for users to gain confidence in them, i.e., their past executions must be auditable so that compliance with, or violations of, policies can be asserted and explained. They also note the similarity between accountability and provenance in scientific experiments. Provenance is a key enabler for accountable systems since it consists of an explicit representation of past processes, which allows us to trace the origin of data, actions and decisions (whether automated or human-driven). It therefore provides the necessary logs to reason about compliance or violations. As users delegate important tasks to systems and endow them with private data, it is crucial that they can put their trust in such systems. Accountability is a way by which trust can be built, since action transparency and audit help users gain trust in systems. However, users may not always want (or have the resources) to audit systems; instead, they would like to be given a measure of trust, which they can rely upon to decide whether to use a system or not.

The output of social machines may be influenced by many factors including collective human input and machine processes. Because information is derived from many agents and processes, it can be challenging to understand and evaluate the results. Provenance information can be used to understand better the results, allowing their reliability and quality to be analysed. Therefore, understanding social computation hinges on capturing the provenance of the creation of data by both humans and machines.

[2] Uniquely identifying participants does not require or imply the use of any personally identifiable information, which would connect the participant to the real person.

3 Use Cases for Provenance, Trust and Reputation

In order develop a fully auditable reputation service, it is necessary to capture provenance information from the applications which use it. Hence, we first consider important design issues and decisions for applying provenance, trust and reputation to social machines. We ground our recommendations in set of generic use cases for social machines. The use cases were outlined by considering generalised user requirements of social machines in the public domain, by investigating a subset of the social machines identified by the SOCIAM and SmartSociety projects [3]. We discuss methods for using provenance in social machines.

We have categorised the use cases into several types: functional, audit, privacy and security, and administration. We note that not all of the use cases are suited to all social machines, and we have described a machine's applicable attributes. We refer to participants as either humans or machines, which take part in the function of the social machine.

3.1 Functional Use Cases

The following use cases describe scenarios where provenance, trust and reputation can be used to support a social machine's functional requirements.

Use Case 1 *A participant creates or edits a piece of information in the system.*

This is the basic use case of such systems, and we require that provenance is captured for the new piece of information created, or the changes to existing information. In social machines like Wikipedia, where the generated information is the actual final output of the system, this kind of information is very important. When an editor changes an article, the information logged comprises of user name or IP address of the editor, the date and time, and the changes made. These information items are made visible to all other participants, passive or active, through the "View history" tab. Depending on how the systems allow access (which based on ownership, access rights, or roles) to the objects they manage, some participants might not be able to edit part of the information, in which situation the next use case is relevant.

Use Case 2 *A participant annotates existing resources in the system.*

An annotation is any meta-information about the main objects used by the social machine. This includes ratings of existing users or products in an online store, feedback on user activities in a listing of service providers, feedback on the quality of data. Information which is the main focus in one social machine, can be considered annotation in another system, for example ratings and comments on products on

[3] A comprehensive list of the systems identified can be found at: http://sociam.org/social-machines

eBay might be annotations, while comments and ratings on TripAdvisor and Yelp are the main focus of the system. If we consider creating annotations as adding new data in the system, then this use case is a sub-part of the previous one.

Use Case 3 *A participant has to make a decision which requires them to select from a subset of the objects (or users) available in the system.*

This use case is relevant to social machines that rely on reputation and user preferences to enable participants to make decisions – make a selection. For example: On Amazon users select which product best suits their needs and rely on the product information and reviews; on TripAdvisor users select hotels or restaurants based on their location preferences, and reviews; and on Stack Overflow users post programming related problems and questions, and receive solutions, and can then select which is the best solution based on their opinion, a voting system, and user reputations.

3.2 Audit Use Cases

The provenance and reputation information collected in a social machine can support the ability to audit it.

Use Case 4 *A participant wants to know who created or changed a resource.*

This use case requires that the system provides a way to expose the users to provenance data captured as a result of the first or second use cases listed above. This use case is applicable to data that has been edited collectively, like for instance Wikipedia articles, where it is important to be able to see when and who made a change. Wikipedia also provides "Talk pages" where edits to articles can be discussed, and which allow participants to understand the motivations behind the changes, so that future edits take into account considerations of past motivations. The next use case refers to annotations, in a similar manner.

Use Case 5 *A participant wants to know who annotated a resource and when.*

Amazon and eBay are social machines whose participants benefit from being able to see provenance information of annotations. The users can decide on the value of ratings by checking who and when they were posted. This will allow the user to make an informed decision about the vote rating.

Use Case 6 *A participant wants to know how the reputation of a user is computed by the system.*

This use case requires that the reputation scores for participants also have provenance information which makes them auditable as well. The method used in computing the reputation scores should be easy to understand and available to participants, as part of the provenance of the reputation. For most online stores, reputation

of sellers is computed in a straightforward manner by averaging the ratings received over time. However, some systems might decide to discard ratings older than a given date, or given by users with a low reputation. Such choices in the formula might improve the overall accuracy of the resulting scores, but they should be known to participants.

Within the scope of this use case is included also the possibility of the user enquiring about their own reputation, to see how they are perceived by other participants in the system and what factors influenced it. This leads to the next use case:

Use Case 7 *A participant has audited their reputation score, but they consider it is incorrect and would like to influence it.*

Some social machines allow users to verify information provided about them by other users, and take under consideration evidence that refutes the incorrect information. An example of this is the eBay Resolution Center, which allows buyers and sellers to resolve conflicts in a controlled environment, before negative ratings are submitted. Yelp on the other hand does not arbitrate reviews, but they do allow businesses to post public responses to reviews, in which to address the issues.

3.3 Privacy and Security Use Cases

The use cases in this section describe scenarios where provenance, trust and reputation can be used to support a user's security and privacy requirements. They are applicable mainly to human users of social machines, especially those systems which request and store personal details, for example Facebook, LinkedIn, Twitter.

Use Case 8 *A participant wants to change their personal information and preferences stored by a social machine.*

This use case includes adding new information, changing existing values, and removing previously set personal data from the system. The users should also be informed what other usage information the social machine captures, and should be able to decide if they agree to this data being stored. For example, Google uses location data from Android phones to map congestion areas in Google Maps[4], but they allow users to opt out of this crowdsourcing experiment.

Use Case 9 *A participant wants to know who has access to their personal information.*

This use case is as much about auditing the system as it is about privacy and security. It includes situations when the user is concerned about who can can see and use their personal information as stored in the system, both among other internal users, and external entities, like advertising companies for example. LinkedIn in particular

[4] http://googleblog.blogspot.co.uk/2009/08/bright-side-of-sitting-in-traffic.html

employs this information to show its users who of its other users has viewed their profile information, as a possible show of interest.

3.4 Administrative Use Cases

The use cases in this section refer to the use of provenance, trust and reputation for the administration of the social machine.

Use Case 10 *An administrator wants to quantify how much of a goal of the system has been achieved.*

This use case is relevant to social machines which have quantifiable goals. It is useful if the overall goal is divided in sub-goals, which must be achieved before the next state or activity can occur. For example, in the CollabMap[5] [33] social machine for map building for emergency response, an evacuation route from a building can only be created after the building outline was created, and validated by a given number of independent users. Some social machines have a running goal, which can never be completed, like for example Wikipedia's aim of capturing world knowledge. Some social machines can have dual goals, one or all of which can be quantifiable, like for instance reCAPTCHA[34] which on one side is used to validate that the user is human by being able to decipher images of words, and at the same time the result is used for digitization of books.

Use Case 11 *An administrator wants to analyse statistics about the users' behaviour and achievements.*

User statistics can be used for tracking the adoption of a social machine, but also for feeding back information into the social machine, and influencing its further development. For example, this use case is applicable to social machines which use gamification elements like star ratings, badges and leader boards. In this case, usage analysis can help to identify behaviours to reward, or how much certain actions should contribute to a user's score, based on provenance. For example, the protein folding game Foldit could analyse provenance data to find who performed complex moves and reward them with a higher score or ranking.

Use Case 12 *An administrator wants to check that the resources are used according to the agreed upon rules.*

This use case is applicable to social machines which require resources to be created in accordance to specific policies. Auditing the provenance data allows the user to validate the sequence of activities performed over entities by an agent.

[5] http://collabmap.org/

4 Designing Provenance, Trust and Reputation in Social Machines

In this section, we present an upper level ontology, Provenance for Social Computations, which defines the schema of provenance concepts in social machines. It is used to define types explained in the provenance, and supports querying and thus auditing. It can also be used to expose provenance to the end user.

Following that, we present a REST API for a reputation service, which stores and retrieves user feedback, and retrieves of reputation information. We use the API to support the capture of provenance for the ride share application, discussed in the following section. It will be used to aid in the development of reusable provenance patterns for REST services.

4.1 Provenance for Social Computation

Provenance is a record of the entities, activities, and agents, that describes the flow of data across a single or multiple systems. In order for provenance to be traceable through heterogeneous systems, which may have their own ways of representing information, it is important to use a shared vocabulary. In order to support heterogeneous social systems, we present the upper level ontology provenance for social computations, which defines a core vocabulary for the classification of agents, entities and activities. Specifically, these three concepts are reused from PROV-O [6], where:

1. A *prov:Entity* is a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary.
2. A *prov:Activity* is something that occurs over a period of time and acts upon or with entities; it may include consuming, processing, transforming, modifying, relocating, using, or generating entities.
3. A *prov:Agent* is something that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent's activity.

The ontology is expressed using the OWL2 Web Ontology Language. It provides a set of classes, properties, and restrictions that can be used to represent and exchange provenance information generated in different systems and under different contexts. It can also be specialised to create new classes and properties to model provenance information for social computations.

→ prov:Agent

→ Collective
- → Machines
- → Users
- → MachinesAndUsers

[6] PROV-O: http://www.w3.org/TR/prov-o/

- → prov:Person
  - → User
    - → AdminUser
    - → GuestUser
    - → LoggedInUser
- → prov:SoftwareAgent
  - → Machine
    - → WebApplication
    - → WebServer
    - → Database

The PROV:Agent class is a parent to three subclasses: prov:Person, prov:SoftwareAgent, and Collectives (see Table 1).

Class	Description
Machine	Identifies software components which run software. Subclasses of Machine include: **WebServer**, **WebApplication** or **Database**.
User	Identifies agents which have a user role within a social computions. A user can be classified as: an **AdminUser** who has administrative privileges; a **GuestUser** where the user is not registered with a social machine; and a **LoggedInUser** who has registered with a user name and password. A GuestUser and LoggedInUser may be the same person at different times, however we chose to differentiate between them because typically social machines allow logged in users different privileges to guest users. While it is important to understand the provenance of an agent, it is also important to be able to describe collectives of agents.
Collective	Identifies a group of agents that are motivated by at least one common issue or interest, or work together to achieve a common objective. The Collective class denotes a group of agents, which can be composed of just one type, like Users or Machines, or a mix of Users and Machines. The notions of the dimensions or characteristics used to define collectives are largely undefined, with respect to social machines. Therefore, the collective subsumption is most likely to evolve with new research efforts. However, the notion of a collective is used when describing or comparing the outcome of a social computation. For example, in GalaxyZoo people classify with collective performance as good as professional astronomers [35].

Table 1: prov:Agent classes and their descriptions

The PROV:Entity class is parent to seven subclasses DataStore, prov:Location, MachineOutput, NegotiationOutcome, Plan, UserInput, and Utility, as shown in Table 2.

- → prov:Entity
  - → DataStore
  - → prov:Location
  - → MachineOutput
  - → Plan
    - → TransformativeInformation
  - → NegotiationOutcome
    - → NegotiationAgreement
    - → NegotiationCounterOffer

Class	Description
DataStore	Identifies data repository entities of a set of integrated objects.
prov:Location	Identifies the geographical location of an entity.
MachineOutput	Defines entities that were created by a machine process, this process may transform the inputs to this activity.
NegotiationOutcome	Defines entities that were produced from a negotiation, where a negation may result in an agreement, rejection or counter offer.
Plan	Defines entities that are a detailed proposal for doing or achieving a goal.
UserInput	Classifies entities that are inputted by users, these may include personal details, user preferences, feedback information such as votes or ratings, or user requests.
Utility	Describes entities that have utility for social computation, such as price.

Table 2: prov:Entity classes and their descriptions

- → UserInput
  - → PersonalDetails
  - → UserPreference
  - → FeedbackInformation
    - → Feedback
    - → Vote
  - → NegotiationInput
  - → UserRequest
- → Utility
  - → Price

The PROV:Activity class is parent to six subclasses (see Table 3):

Class	Description
PerformNegotiation	Describes negotiation activities.
ProvideInformation	Describes activities that provide a prov:Entity.
CompleteTask	Describes activities that are performed by a user or machine and may result in Plan, Utility, UserInput, and or Negotiation entities.
PublishData	Describes activities which can be performed by a user or machine who publishes data, a user may publish their contribution, and a machine may publish the result of a computation.
RunPlan	Describes activities which can be performed by a user or a machine
StoreData	Describes activities which store data.

Table 3: prov:Activity classes and their descriptions

- → prov:Activity
  - → PerformNegotiation
    - → SubmitApproval
    - → SubmitCounterOffer
    - → SubmitDisagreement
  - → ProvideInformation
    - → ProvideFeedback

- → ProvideVote
- → ProvideStarRating
- → CompleteTask
  - → CompleteMirotask
  - → PlayGame
- → CreateOriginalContent
- → ProvideDeviceCollectedData

→ PublishData
→ RunPlan
- → PerformDataManipuation
  - → Additive
  - → Subtactive
  - → Transformative
- → PerformHouseKeeping

→ StoreData
→ RetrieveData

Applications may extend this ontology with their own class hierarchy, because the aim of the upper level ontology is to capture the core concepts and their properties. An example of this is described in Section 5, where we extend the ontology with concepts specific to the ride share application.

4.2 Trust and Reputation

Social machines, such as LinkedIn, Stack Overflow, and eBay, use reputations to allow users to make trust judgements, and also to instil trust in the system. A reputation service will compute and publish reputation scores for a set of subjects (such as users, goods, service providers and services) within a social machine, based on the opinions of other users about a subject. The opinions are typically ratings and are sent to the reputation service, which uses a specific reputation algorithm to dynamically compute the reputation scores based on the received ratings.

Users of a social machine use the reputation scores for decision making: a subject with a high reputation score will normally attract more business than a subject with a low reputation score. It is therefore in the interest of users to: have a high reputation score; know what factors influenced the reputation score; and understand how to improve their score. Also, a transparent reputation system, which is clear in the way it computes scores, appears more trustworthy to its users.

In order to allow social machines to provide reputation data and access the reputations, we contribute a RESTful API because it helps organise a complex application into simple resources which it makes it easy for new clients to use the application, even if it is not specifically designed for them. The following REST API described in Table 4, has four resources: subjects, feedback reports, reputation reports, and events. A subject is the subject about which feedback or reputation describes, and is derived from feedback reports provided by an author. A feedback report can be associated with an event, which a subject was a part of. In more detail, an event is identified with a time and date range to which the feedback is pertaining to.

Action	Description
GET /subjects/	Get the URIs of subjects which have reputations.
GET /subjects/:subject/feedback-reports/	Get the URIs of the feedback reports about the subject.
GET /subjects/:subject/reputation-reports/	Get the URIs of the reputation reports about the subject.
GET /subjects/:subject/feedback-reports/:report/	Get a feedback with a report identifier about the subject.
GET /subjects/:subject/reputation-reports/:report/	Get a reputation report with a report identifier about the subject.
POST /subjects/:subject/feedback-reports/	Post a new reputation report about the subject.
GET /subjects/:subject/events/	Gets the URIs of events a subject is associated with (for example, in the ride share application users are associated with ride request id).
GET /subjects/:subject/feedback-reports/?event=:event	Gets the URIs of the feedback reports about the subject from an event.
GET /subjects/:subject/feedback-reports/?author=:user	Get the feedback reports that is authored by a user about a subject.
GET /subjects/:subject/reputation-reports/summary-latest	Get latest reputation report, which is the latest generated summary from the reputation service about a user.

Table 4: The Reputation Service's URIs

5 The Ride Share Application

This section describes a ride sharing application that allows drivers and commuters to offer and make request rides [36]. These offers and ride requests include details about required travels, timing, locations, capacity, prices, and other details relevant for car sharing. It performs automatic matching of commuters to available cars, by considering departure and destination locations, routes, capacity and reputation. The interactions of a driver and commuters differ and result in different outcomes. The following list describes the flow of interactions when a driver offers a ride.

1. Drivers and Commuters post *ride requests* to the server.
2. Matching is performed and some potential ride plans are generated based on the previously submitted ride requests from commuters that are matching the constraints of the ride request posted by the driver. This gives rise to *ride plans* that appear as potential ride plans both for the driver as well as for the commuters who have already submitted ride requests in the past, these ride requests have not been finalised (i.e. a ride record does not exist for them) and are matched to the ride request of the driver.

3. When at least one driver or commuter indicates their willingness to follow a specific ride plan the specific potential ride plan becomes a *potentially agreed ride plan*.
4. When all participants have expressed an interest in a potentially agreed ride plan, the driver selects one and attempts to finalise negotiation.
5. When all the commuters who appear in the driver agreed ride plan also agree that this is their selection among their driver agreed ride plans, the agreement has been reached and this gives rise to an *agreed ride plan*.
6. Together with the agreed ride and all the other ride plans that exist, both for the driver as well as the commuters, automatically become invalid ride plans for the specific requests that generated them.

5.1 The Ride Sharing Architecture

The ride share application has five core components: a view, ride matching service, reputation service, and ProvStore [7] (see Figure 2). The view provides the user with the graphical components with which to enter their ride requests, and to view and select potential rides. The matching service provides matches containing drivers and commuters, which the users can select. The reputation service is designed to store feedback reports and, generate and store reputation reports. The ProvStore is a specialised service for storing provenance using W3C PROV standard, the view, matching service and reputation service all send provenance data to it.

The components communicate using REST APIs. In more detail, we show the interactions between the ride share service, reputation service and the ProvStore in Figure 3. The interactions use a REST API for the reputation and ProvStore services. The figure describes five interactions:

1. The ride share application requests the latest reputation of a user. It sends a GET request for the latest generated reputation of the user, and receives a JSON object containing the reputation. The reputation service also generates the provenance data recording this request and the outcome, and posts it to the ProvStore. This interaction may occur when a user requests to view the reputation of another, and the ride matching algorithm filters the potential rides.
2. The ride share application requests all feedback reports about a subject authored by a given author. This interaction occurs when the ride manager is matching drivers and users for rides, if an author rates the other participant (the subject) highly then this is more likely to result in a match. The reputation service returns a JSON object containing the requested feedback reports, and sends the provenance data recording this request to the ProvStore.
3. The ride share application requests the feedback reports about a user, the ride share application sends two requests. The first GET requests returns the dictionary of reports describing a user, and the second GET requests for a particular

[7] ProvStore: https://provenance.ecs.soton.ac.uk/store/

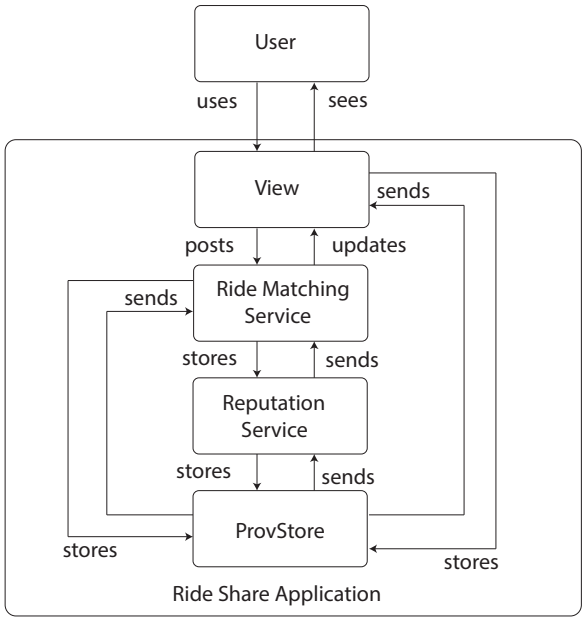


Fig. 2: Components of the ride share application

report. The reputation service sends the provenance data recording this request to the ProvStore.

4. The ride share application submits a feedback report. The reputation service stores the feedback report, and then generates a reputation report of the user. In order to generate the reputation of a user, it requests details from the ProvStore about the user. The reputation service posts the provenance data recording this request to the ProvStore.
5. The ride share application submit provenance data to the ProvStore. The ride share application posts the provenance data contained in a bundle to the ProvStore. This interaction occurs when the ride share application creates entities and performs activities on entities, such as receiving ride requests from users and generating ride plans.

5.2 Provenance Example for the Ride Share Application

In order to describe the provenance generated by the ride share application, we run through an example where two users post ride requests and describe the provenance generated in each step.

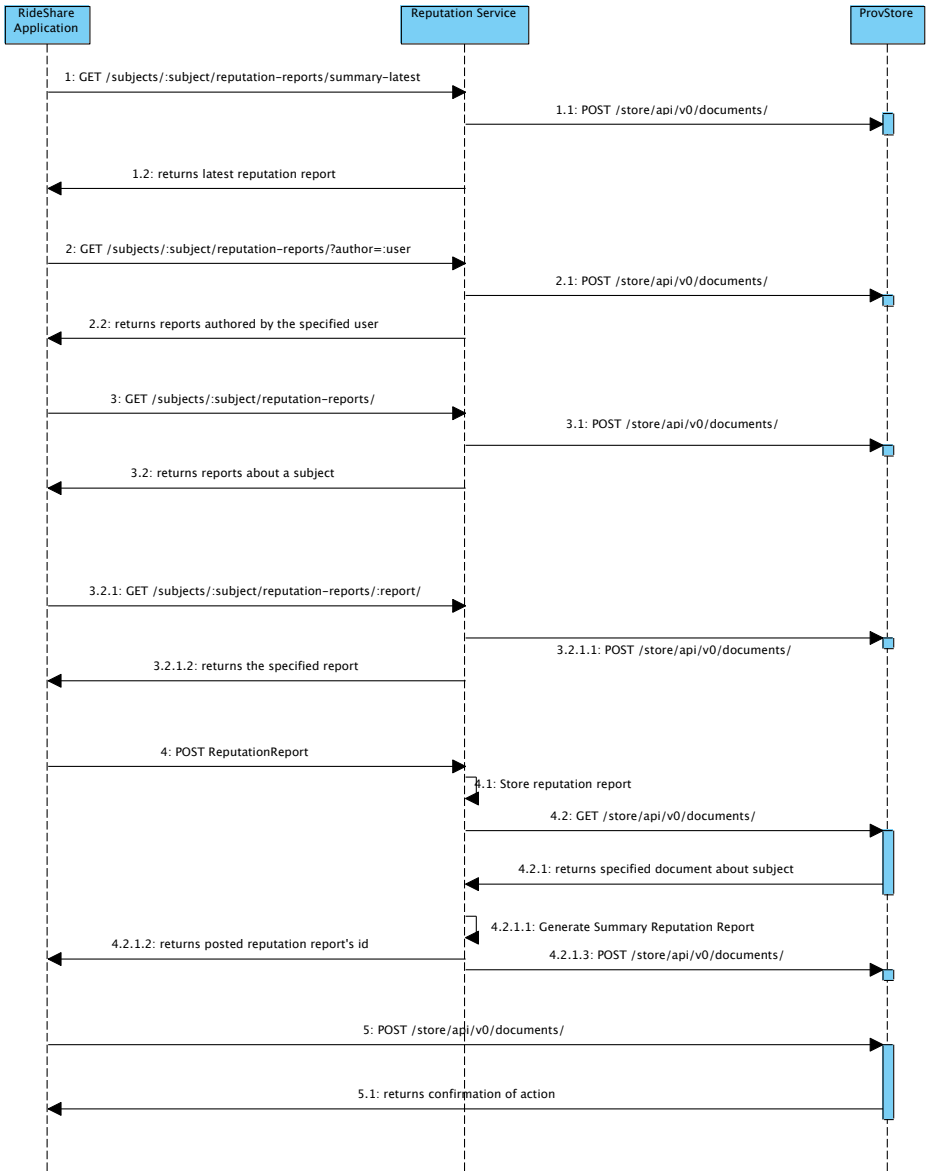


Fig. 3: Interactions between the ride share application, reputation service and ProvStore

1. Alice who is a driver wants to car pool because it saves her money on petrol, she uses the ride share application to find someone to share a ride with. She logs into the application and creates a ride request with the details of the ride, including the departure and destination locations and times, how many seats are available and her preferences.
   The provenance recorded from posting a ride request is shown in Figure 4. Alice who is identifiable by the uri *rs:users/0*, posted the ride request *rs:rideRequest/#0*[8] to the *ride_manager*, who stored the ride request at *rs:rideRequest/0*.

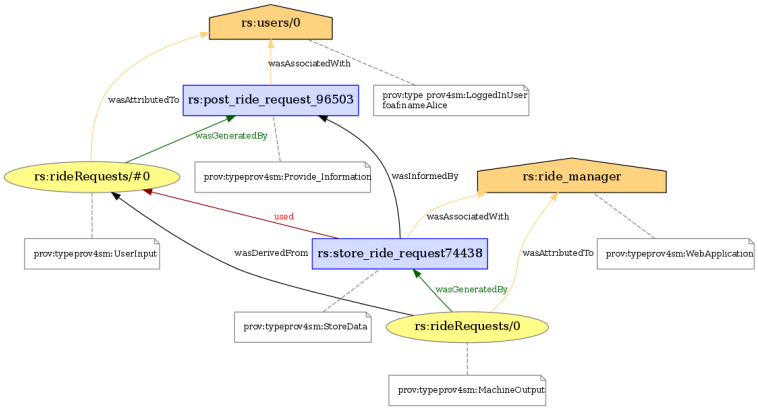
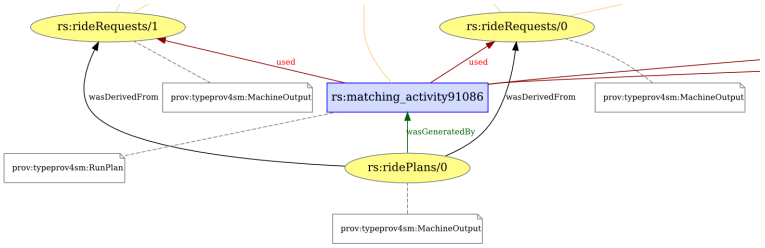


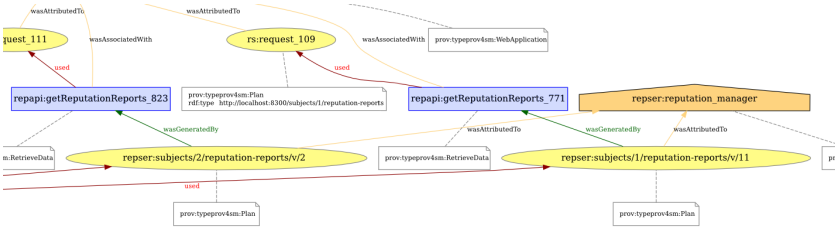
Fig. 4: A graph showing the provenance generated by Alice's ride request.

2. Bob would like a ride, and he logs in to the ride share application and adds his request to the application.
   The provenance recorded Bob, who is identifiable the uri *rs:users/1* posting the ride request *rs:rideRequest/#1* to the *ride_manager*, who stored the ride request at *rs:rideRequest/1*.
3. After each ride request is submitted, the ride share application runs a ride matching algorithm. It uses the information in the two ride requests Alice and Bob submitted because their departure and destination locations and time were compatible, it also used their reputation which was stored by the ride manager, to generate a match.
   The provenance recorded from this step shows that the matching algorithm used Alice and Bob's ride requests, and their reputations which are retrieved from the *reputation manager*, to generate the ride plan, *rs:ridePlans/0*. This is shown in Figures 5a and 5b.
4. Alice and Bob can then view that there is a match. Alice views Bob's reputation and then accepts the ride.

[8] The hash indicates that this entity isn't stored in memory.



(a) The matching algorithm used Alice and Bob's ride requests.



(b) The matching algorithm used Alice and Bob's reputation, and generated a ride plan.

Fig. 5: A detail of the provenance generated by the ride matching activity.

This provenance generated by Alice's acceptance is shown in Figure 6. It shows that Alice viewed Bob's reputation *repser:subjects/28/reputation-reports/v/11* and accepted the ride plan *rs:ridePlans/0*, which changed the state of the ride plan into a potentially agreed ride plan *rs:ridePlan/0/v/parp*.

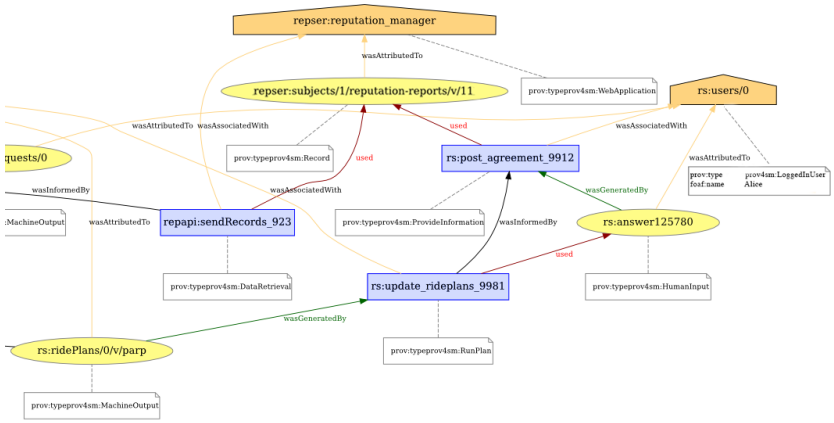
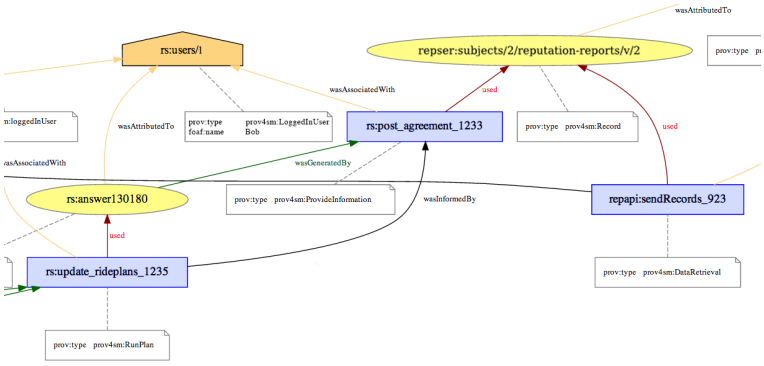
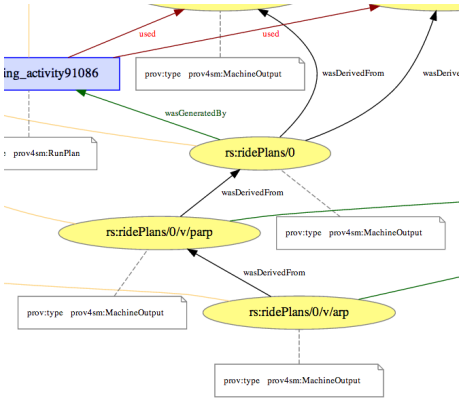


Fig. 6: A detail of the provenance generated by Alice's acceptance of a ride plan.

5. Bob then views Alice's reputation, and accepts the ride.
   The provenance that was generated by this acceptance shows that Bob viewed Alice's reputation *repser:subjects/27/reputation-reports/v/2* and accepted the ride plan *rs:ridePlan/0/v/parp*, which changed the state of the ride plan into an agreed ride plan *rs:ridePlan/0/v/arp*, as shown in Figures 7a and 7b.



(a) Bob viewed Alice's reputation



(b) The agree plan was generated.

Fig. 7: Details of the provenance generated by Alice's acceptance of a ride plan.

6. When the ride has been completed, Bob fills in a feedback form rating Alice as a Driver. This triggers the reputation manager to recalculate Alice's reputation. The provenance generated by the submission of Bob's feedback, shows that the *rs:ride_manager* posts a reputation report *repser:subjects/27/reputation-reports/#490* and is used to generate a reputation report with the activity *repapi:generateReputationReports_1244*. This activity generates the reputation report *repser:subjects/27/reputation-report/491*, which is derived

from the properties (such as the *#total_completed_rides*) in the report *repser:subjects/27/reputation-reports/490* (see Figure 8).

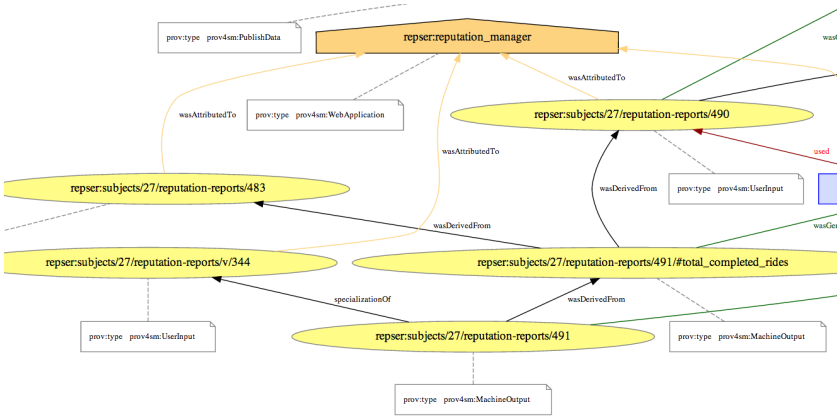


Fig. 8: A detail of the provenance generated when Bob submits feedback about Alice.

The reputation report *repser:subjects/27/reputation-report/491* contains Alice's reputation in JSON format shown in Table 5. The reputation report includes a *hasProvenance* property, which is a uri that links to the provenance of the reputation report.

5.3 Accountable Reputation Service

The provenance recorded from the above steps allows users to perform audits, including:

1. Who created ride requests;
2. Which ride requests were used to generate which ride plans;
3. Who accepted and rejected ride plans;
4. How users' rides are generated and what influenced their generation;
5. How users' reputations were generated.

Moreover, it allows the users to overview the provenance of more than one service, the ride and reputation manager, which can be difficult in heterogeneous systems. It also gives the users the awareness that their actions are accountable, and it allows users to alter their actions so that they might improve their chances of ride matches and being selected by other users. Specifically, it supports the following use cases, which are derived from the general use cases presented in Section 3:

```
{
    ”report_type” : ”reputationReport”,
    ”user_id” : 0”,
    ”total-stars”: 321,
    ”total_completed_rides”: 201,
    ”number_of_repeat_riders” 82,
    ”average_overallStarRating”: 4.5,
    ”average_ride_Price”: 3,
    ”average_ride_Route”: 4,
    ”average_ride_Car/Environment”: 2,
    ”average_ride_OnTime”: 5,
    ”average_individual_Reliability”: 5,
    ”average_individual_Communication”: 4,
    ”average_individual_DrivingSkill”: 3,
    ”average_individual_Friendliness”: 4,
    ”average_outsideFactors_Traffic”: 2,
    ”average_outsideFactors_Weather”: 2,
    ”freeTextComments” : [”Problems with traffic but other-
    wise fine.],
    ”hasProvenance” : ”https://provenance.ecs.soton.ac.uk/store/documents/1665/”,
    ”provenanceGenerated” : ”2014-01-30T21:00:00.250”
}
```


Table 5: Alice's reputation report *repser:subjects/27/reputation-report/491*, which was derived from Bob's feedback report.

Ride Share Use Case 1 *The user wants to be able to make choices between available rides based on the participants and their preferences, reputation, and their opinion of them. This use case is a specialisation of use case 3.*

Ride Share Use Case 2 *The user wants to be able to analyse quickly the possible participants, and if the choice is too large then they should be filtered to include only rides that fulfil their preferences with users that have good reputations. This use case is a specialisation of use case 3.*

Ride Share Use Case 3 *The user wants to be able to understand why they were recommended particular ride matches, so that they can see which factors affected the recommendation, such as their preferences or reputation. This use case is a specialisation of use case 3.*

Ride Share Use Case 4 *The user wants to be able to understand how they are viewed by others in the ride share application, and which factors influenced this. This use case is a specialisation of use case 6.*

Ride Share Use Case 5 *The user wants to understand how their personal details are used by the ride share application system. This use case is a specialisation of use case 9.*

6 Summary

Provenance describes the flow of data across multiple systems, as we have demonstrated in the previous section with the ride and reputation manager. Moreover, provenance is independent of the technologies used in those systems executions. This is crucial because heterogeneous systems implemented by different developers or companies, and may each have their own way of representing information. In order to support heterogeneous social systems, we present the upper level ontology provenance for social computations, which defines a core vocabulary for the classification of agents, entities and activities.

Auditing private data processing is crucial so that authorities and system administrators can check its compliance with regulations. Provenance is a record of data entities, and it details how entities are created, modified and used, and this record can be used to audit these processes. In order to consume the provenance data generated by a social machine it must be retrievable.

Exposing provenance data increases public awareness and promotes accountability. Organisations are required to manage personal information in accordance with regulatory frameworks. However, there have been several cases where personal information has been leaked and exposed to unauthorised recipients. Future work will explore how to expose provenance without revealing the identity of users.

Reputation plays a core role in social machines, directly by interactions between users of social machines, and indirectly by influencing social computations. Therefore, in this paper we provide a generic REST API for exposing reputation information.

7 Acknowledgements

We would like to acknowledge the SmartSociety and SOCIAM projects. The SmartSociety Project is funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n 600854. SOCIAM: The Theory and Practice of Social Machines. The SOCIAM Project is funded by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant number EP/J017728/1 and comprises the Universities of Southampton, Oxford and Edinburgh. The authors would also like to acknowledge Avi Segal, Dimitrios I. Diochnos, Kevin Page, Kobi Gal and Michael Rovatsos, for their work on the ride share application.

References

1. D.J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, G.J. Sussman, (ACM, 2008), pp. 82–87

2. T.D. Huynh. Trust and reputation in open multi-agent systems (2006). URL http://eprints.soton.ac.uk/262759/
3. L. Moreau, Web Semantics: Science Services and Agents on the World Wide Web **9**(2), 202 (2011). URL http://eprints.soton.ac.uk/271992/
4. L. Moreau, P. Missier (eds.), K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker, S. Miles, J. Myers, S. Sahoo, C. Tilmes, PROV-DM: The PROV Data Model. W3C Recommendation REC-prov-dm-20130430, World Wide Web Consortium (2013). URL http://www.w3.org/TR/2013/REC-prov-dm-20130430/
5. T. Lebo, S. Sahoo, D. McGuinness (eds.), K. Behajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik, J. Zhao, PROV-O: The PROV Ontology. W3C Recommendation REC-prov-o-20130430, World Wide Web Consortium (2013). URL http://www.w3.org/TR/2013/REC-prov-o-20130430/
6. H. Hua, C. Tilmes, S. Zednik (eds.), L. Moreau, PROV-XML: The PROV XML Schema. W3C Working Group Note NOTE-prov-xml-20130430, World Wide Web Consortium (2013). URL http://www.w3.org/TR/2013/NOTE-prov-xml-20130430/
7. L. Moreau, P. Missier (eds.), J. Cheney, S. Soiland-Reyes, PROV-N: The Provenance Notation. W3C Recommendation REC-prov-n-20130430, World Wide Web Consortium (2013). URL http://www.w3.org/TR/2013/REC-prov-n-20130430/
8. L. Moreau, P. Groth, *Provenance: An Introduction to PROV* (Morgan and Claypool, 2013). URL http://dx.doi.org/10.2200/S00528ED1V01Y201308WBE007
9. S.D. Ramchurn, D. Huynh, N.R. Jennings, et al., The Knowledge Engineering Review **19**(1), 1 (2004)
10. J. Sabater, C. Sierra, Artificial Intelligence Review **24**(1), 33 (2005). DOI 10.1007/s10462-004-0041-5. URL http://dx.doi.org/10.1007/s10462-004-0041-5
11. D. Artz, Y. Gil, Web Semantics: Science, Services and Agents on the World Wide Web **5**(2), 58 (2007)
12. J. Golbeck, in *Computing with Social Trust* (Springer London, 2009), pp. 1–5
13. I. Pinyol, J. Sabater-Mir, Artificial Intelligence Review **40**(1), 1 (2013). DOI 10.1007/s10462-011-9277-z. URL http://dx.doi.org/10.1007/s10462-011-9277-z
14. J. Golbeck, J. Hendler, in *Engineering Knowledge in the Age of the Semantic Web, Proceedings of the 14th International Conference, EKAW 2004*, *Lecture Notes in Computer Science*, vol. 3257, ed. by E. Motta, N.R. Shadbolt, A. Stutt, N. Gibbins (Springer Berlin Heidelberg, 2004), *Lecture Notes in Computer Science*, vol. 3257, pp. 116–131. DOI 10.1007/978-3-540-30202-5_8. URL http://dx.doi.org/10.1007/978-3-540-30202-5_8
15. J.A. Golbeck, Computing and applying trust in web-based social networks. Ph.D. thesis, College Park, MD, USA (2005). AAI3178583
16. J. Golbeck, Found. Trends Web Sci. **1**(2), 131 (2006). DOI 10.1561/1800000006. URL http://dx.doi.org/10.1561/1800000006
17. B. Huang, A. Kimmig, L. Getoor, J. Golbeck, in *Social Computing, Behavioral-Cultural Modeling and Prediction* (Springer Berlin Heidelberg, 2013), pp. 265–273
18. T. Berners-Lee, M. Fischetti, M.L. Foreword By-Dertouzos, *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor* (HarperInformation, 2000)
19. T. Berners-Lee, Design Issues: Cleaning up the User Interface. W3C Note, World Wide Web Consortium (1997). URL http://www.w3.org/DesignIssues/UI.html
20. X. Li, T. Lebo, D.L. McGuinness, in *Provenance and Annotation of Data and Processes*, *Lecture Notes in Computer Science*, vol. 6378, ed. by D.L. McGuinness, J. Michaelis, L. Moreau (Springer Berlin Heidelberg, 2010), pp. 182–197. DOI 10.1007/978-3-642-17819-1_21. URL http://dx.doi.org/10.1007/978-3-642-17819-1_21
21. N. Prat, S. Madnick, in *Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences* (IEEE Computer Society, Washington, DC, USA, 2008), HICSS '08, pp. 393–. DOI 10.1109/HICSS.2008.243. URL http://dx.doi.org/10.1109/HICSS.2008.243

22. I. Pinyol, J. Sabater-Mir, International Journal of Approximate Reasoning **54**(5), 667 (2013). DOI http://dx.doi.org/10.1016/j.ijar.2012.11.006. URL http://www.sciencedirect.com/science/article/pii/S0888613X1200196X
23. Y. Gil, D. Artz, Web Semantics: Science, Services and Agents on the World Wide Web **5**(4), 227 (2007). DOI http://dx.doi.org/10.1016/j.websem.2007.09.005. URL http://www.sciencedirect.com/science/article/pii/S1570826807000376. ¡ce:title¿World Wide Web Conference 2006Semantic Web Track¡/ce:title¿
24. M. Richardson, R. Agrawal, P. Domingos, in *The Semantic Web - ISWC 2003*, *Lecture Notes in Computer Science*, vol. 2870, ed. by D. Fensel, K. Sycara, J. Mylopoulos (Springer Berlin Heidelberg, 2003), *Lecture Notes in Computer Science*, vol. 2870, pp. 351–368. DOI 10.1007/978-3-540-39718-2_23. URL http://dx.doi.org/10.1007/978-3-540-39718-2_23
25. A. Jøsang, R. Ismail, C. Boyd, Decis. Support Syst. **43**(2), 618 (2007). DOI 10.1016/j.dss.2005.05.019. URL http://dx.doi.org/10.1016/j.dss.2005.05.019
26. P.A. Chirita, W. Nejdl, M.T. Schlosser, O. Scurtu, in *ISWC Workshop on Trust, Security, and Reputation on the Semantic Web* (2004)
27. E. Koutrouli, A. Tsalgatidou, in *Proceedings of the Third International Conference on Trust, Privacy, and Security in Digital Business* (Springer-Verlag, Berlin, Heidelberg, 2006), TrustBus'06, pp. 152–161. DOI 10.1007/11824633_16. URL http://dx.doi.org/10.1007/11824633_16
28. L.H. Vu, K. Aberer, ACM Trans. Auton. Adapt. Syst. **6**(4), 24:1 (2011). DOI 10.1145/2019591.2019593. URL http://doi.acm.org/10.1145/2019591.2019593
29. L. Liu, M. Munro, Decis. Support Syst. **52**(2), 438 (2012). DOI 10.1016/j.dss.2011.10.003. URL http://dx.doi.org/10.1016/j.dss.2011.10.003
30. P. Smart, E. Simperl, N. Shadbolt, in *Social Collective Intelligence* (2014)
31. N.R. Shadbolt, D.A. Smith, E. Simperl, M.V. Kleek, Y. Yang, W. Hall, in *SOCM2013: The Theory and Practice of Social Machines* (2013). URL http://eprints.soton.ac.uk/350513/
32. K. Stein, C. Hess, in *Proceedings of the Eighteenth Conference on Hypertext and Hypermedia* (ACM, New York, NY, USA, 2007), HT '07, pp. 171–174. DOI 10.1145/1286240.1286290. URL http://doi.acm.org/10.1145/1286240.1286290
33. S.D. Ramchurn, T.D. Huynh, M. Venanzi, B. Shi, in *The 5th Annual ACM Web Science Conference* (2013), pp. 326–335. URL http://eprints.soton.ac.uk/350677/
34. L. von Ahn, B. Maurer, C. Mcmillen, D. Abraham, M. Blum, Science **321**(5895), 1465 (2008). URL http://dx.doi.org/10.1126/science.1160379
35. N.R. Shadbolt, D.A. Smith, E. Simperl, M. Van Kleek, Y. Yang, W. Hall, in *Proceedings of the 22nd international conference on World Wide Web companion* (International World Wide Web Conferences Steering Committee, 2013), pp. 905–912
36. K. Gal, A. Segal, D.I. Diochnos, K. Page, H.S. Packer, L. Moreau, M. Rovatsos, Rideshare: A smart society application. Tech. rep., University of Ben Gurion, University of Edinburgh, and University of Southampton (2014). URL https://docs.google.com/document/d/1cNhS-sW8pVuG5XYzrn8R-NeKrzGszkDYJ3KwKQ/edit?invite=CPnQk7sG&pil=1