2012

Trust and Reputation for Successful Software Self-Organisation

Pierpaolo Dondio
Technological University Dublin, pierpaolo.dondio@tudublin.ie

Jean Marc Seigneur
University of Geneva

Recommended Citation

Seigneur,J, M. & Dondio, P. (2011). Trust and Reputation for Successful Software Self-Organisation, in Di Marzo Serugendo, Giovanna, Marie-Pierre Gleizes, and Anthony Karageorgos.(eds) *Self-organising Software: From Natural to Artificial Adaptation*. Springer. doi:10.1007/978-3-642-17348-6_8

Chapter 8
Trust and Reputation for Successful Software Self-Organisation

Jean-Marc Seigneur and Pierpaolo Dondio

More and more software is reused, mixed and mingled. How to obtain a trustworthy software mix?

Abstract An increasing number of dynamic software evolution approaches is commonly based on integrating or utilising new pieces of software. This requires resolution of issues such as ensuring awareness of newly available software pieces and selection of most appropriate software pieces to use. Other chapters in this book discuss dynamic software evolution focusing primarily on awareness, integration and utilisation of new software pieces, paying less attention on how selection among different software pieces is made. The selection issue is quite important since in the increasingly dynamic software world quite a few new software pieces occur over time, some of which being of lower utility, lower quality or even potentially harmful and malicious (for example, a new piece of software may contain hidden spyware or it may be a virus). In this chapter, we describe how computational trust and reputation can be used to avoid choosing new pieces of software that may be malicious or of lower quality. We start by describing computational models of trust and reputation and subsequently we apply them in two application domains. Firstly, in quality assessment of open source software, discussing the case where different trustors have different understandings of trust and trust estimation methods. Secondly, in protection of open collaborative software, such as Wikipedia.

Objectives After reading this chapter the reader will:

- understand what computational trust and reputation are;

Jean-Marc Seigneur
University of Geneva, Geneva, Switzerland e-mail: Jean-Marc.Seigneur@trustcomp.org

Pierpaolo Dondio
Trinity College Dublin, Dublin, Ireland e-mail: dondiop@cs.tcd.ie

- know how computational trust and reputation can be used for new software piece selection, even when trustors have different prespectives regarding trust definition and estimation methods;
- understand the practical issues of applying computational trust and reputation, especially in two major applications domains: open-source software quality assessment and open collaborative authoring software protection.

8.1 Background Context

In the human world, trust can exist between two interacting entities and it can be very useful when there is uncertainty involved with the interaction result. The requested entity uses the level of trust it has developed for the requesting entity as a means to cope with uncertainty, for example to engage in actions that involve risks of a harmful outcome. There are numerous definitions of the human notion trust in a wide range of domains, with different approaches and methodologies such as sociology, psychology, economics, and pedagogy. These definitions may even change when the application domain changes. However, it has been convincingly argued that these divergent trust definitions can fit together[371]. Romano's recent definition [445] tries to encompass the previous work in all domains:

> ...**trust** is a subjective assessment of another's influence in terms of the extent of one's perceptions about the quality and significance of another's impact over one's outcomes in a given situation, such that one's expectation of, openness to, and inclination toward such influence provide a sense of control over the potential outcomes of the situation.

Interactions with uncertain result between entities also happen in the online world. For example if one provides open Wi-Fi network access to passing by strangers there is a possibility that they will use the Wi-Fi connection maliciously. Therefore it can be useful to take decisions based on trust in the online world as well. However, the terms "trust", "trusted", "trustworthy" and the like, which appear in the traditional computer science literature, have rarely been based on comprehensive multi-disciplinary trust models and they often correspond to an implicit trust concept, a limited view of the facetted human notion of trust. [69] coined the term *decentralised trust management* because their approach separates trust management from application: their PolicyMaker model introduced the fundamental concepts of policy, credential, and trust relationship. [493] argued that decentralised trust management still relies on an implicit notion of trust because it only describes:

> ...a way of exploiting established trust relationships for distributed security policy management without determining how these relationships are formed.

8.2 Theoretical Notions

A computational model of trust based on social research was first proposed in [356]. Usually, a *trustor* is the user who considers trusting another user, a *trustee*. A third-user may recommend the trustee to the trustor based on a recommendation previously received from elsewhere. Sometimes it may not be known who have made recommendations about a trustee. In such cases, where the identity of the recommenders is not exactly known, the resulting overall recommendation that can be taken into account by a trustor is refered to as the *reputation* of the trustee. In social research, there are three main types of trust: **interpersonal trust**, based on past interactions with the trustee; **dispositional trust**, provided by the trustor's general disposition towards trust, independently of the trustee; and **system trust**, provided by external means such as insurance or laws [371, 432]. Trust concerning a particular situation or context is termed *trust context*. Each trust context is assigned an importance value in the range [0,1] and utility value in the range [-1,1]. The utility is the utility of the action to be allowed if the trustor chooses to trust the trustee and allow her to carry out the action. Any trust value is in the range [-1,1). In addition, each virtual identity is assigned a general trust value, which is based on all the trust values with this virtual identity in all the trust contexts. Dispositional trust appears in the model as the basic trust value: it is the total trust values in all contexts in all virtual identities with whom the trustor has interacted so far. Risk is used in calculating a threshold for trusting decision making.

Generally, a computed trust value of an entity can be seen as the digital representation of the trustworthiness or level of trust of that entity. The trustcomp online community ("Trustcomp") [9] defines entiTrust (to emphasise that it cannot correspond exactly to real-world trust and avoid that users abstract it to their real-world expectation of trust) as a non-enforceable estimate of the entity's future behaviour in a given context based on past evidence. The EU-funded (SECURE) [466] project represents an example of a trust engine that uses evidence to compute trust values in entities and corresponds to evidence-based trust management systems (see Fig. 8.1 below). Evidence encompasses outcome observations, recommendations and reputation. A trust metric consists of the different computations and communications which are carried out by the trustor (and his/her network) to compute a trust value in the trustee. Furthermore, [288] remark that:

> . . .direct experiences and witness information are the "traditional" information sources used by computational trust and reputation models.

Depending on the application domain, particular types of evidence may be more weighted in trust computation than others. For example, in a scenario where all potential recommenders are known by the user in advance, such as when friends in a social network have been manually specified by the user, recommendations can be weighted based on the user's direct observations. When recommendations are processed, the social network can be reconstructed. Along this line, [221] studied the problem of propagating trust value in social networks and proposed an exten-

sion to the FOAF vocabulary[1] together with suitable algorithms for propagating user-estimated trust values instead of only computer-calculated ones. A more efficient algorithm for trust and recommendations propagation in peer-to-peer networks has later been proposed in [146]. Recent approaches base trust value computation on new types of evidence. For example, [561] have discovered an interesting correlation between similarity and trust between social network users, indicating that similarity may be evidence of trust. However, as is the case for trust values that are manually set, it is difficult to accurately estimate user similarity based on a specific and generally applicable set of pieces of evidence. Therefore, trust values are quite often computed from evidence of different types depending on the application domain. Although most work in this direction has so far focused on counting interaction outcomes, other types of evidence may also be found. This approach does not contradict the high level view of the trust engine depicted in Fig. 8.1 because any type of evidence can be stored in the evidence store for future trust calculation.

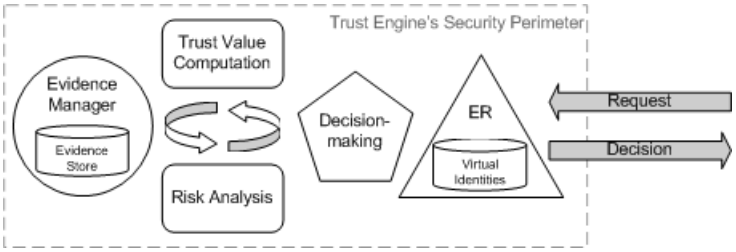

Fig. 8.1 High-level View of a Trust Engine

8.2.1 Evidence-based Trust Engine

In Fig. 8.1, the decision-making component can be invoked when a trusting decision has to be made. The most common scenario considered in the relevant research works includes a requested entity having to decide the next action to be taken after a request made by another entity, the requesting entity. For this reason a specific module termed Entity Recognition (ER) [466] is required to recognise any entities and their associated virtual identities, and to handle their requests. The decision-making component comprises two sub-components:

- a trust module that can dynamically assess the trustworthiness of the requesting entity based on the trust evidence of any type stored in the evidence store;

[1] FOAF (Friend of a Friend) is a project devoted to linking people and information using the Web. The FOAF vocabulary is a machine-readable ontology describing persons, their activities and their relations to other people and objects, which is defined using RDF and OWL technologies.

- a risk module that can dynamically evaluate the risk involved in the interaction, again based on the available evidence in the evidence store.

A common decision-making policy is to select (or recommend to the user) an action that would maintain an appropriate cost/benefit ratio. The Evidence Manager component is responsible for gathering evidence, such as recommendations, comparisons between expected outcomes of the chosen actions and real outcomes. This evidence is used to update risk and trust evidence. Thus, trust and risk follow a managed life-cycle. Given that new types of trust evidence may still be found, it is challenging to go beyond this high-level view of a trust engine, that is, to propose a generic implementation of a trust engine that would work for any application domain. The SECURE trust engine has been an attempt in this direction but evidence such as similarity between users or manually defined user trust values without a clear count of evidence have not been considered yet.

Other trust engines have been designed for specific application domains. For example, the TriQL.P Trust Architecture [68] aims at supporting users in their decision whether to trust or to distrust information found on the Semantic Web. The main types of evidence are the context, which includes who and when, and content, which is related to similarity, such as the inferred main topic of two Web pages. An additional example is Jøsang's computational trust approach termed "subjective logic" [289]. That approach integrates the elements of ignorance and uncertainty, which cannot be reflected by mere probabilities and are part of the human aspect of trust. To represent imperfect knowledge, an opinion is modelled as a triplet whose elements are: $belief(b)$, $disbelief(d)$ and $uncertainty(u)$.

The subjective logic provides more than ten operators to combine opinions. For example, the recommendation operator adjusts the recommended opinion based on the recommending trustworthiness (RT) parameter. Jøsang's approach can be used in many applications since the trust context is open. However, it is still limited to few trust evidence types, such as direct observations of outcomes or recommendations. In addition, there is no risk component. [99] argue for a trust engine based on cognitive science where the main trust evidence types originate from the entity's belief and goals structure instead of probabilistic quantitative views, economics or game theory. Furthermore, [231] claim to have built a generic open rating system, which means that anybody is allowed to rate anything in the system, including the ratings of contents. In addition, [323] highlighted the impact of trust in expert systems advice. Subsequently, [36] proposed an expert system containing knowledge about the factors taken into account to compute trust in certification authorities participating in in public key infrastructures. In that case, the trust engine is merely mapped to a generic expert system where emphasis is on the knowledge of the particular application domain incorporated in the system by human experts.

8.2.2 Computational Trust Methodology Overview

Based on the previous work surveyed above, the informal methodology that has generally been used to apply computational trust is as follows:

1. a model of trust from previous multi-disciplinary work on the human notion of trust is reused or refined to be turned into a computational model;
2. the main types of trust evidence relevant to the application domain are given more weight;
3. a computational version of the trust model is deployed and evidence is collected and fed in the model using the calculated trust to handle uncertainty.

The first of the above steps has sucessfuly been applied in specific application domains such as peer-to-peer file sharing, and the basic types of trust evidence considered, such as positive and negative downloads count, provided satisfactory results for simple cases. As [288] underlined:

> ...game theoretical models have given good results in simple scenarios, but may be too limited for more complex scenarios.

Further to the above mentioned informal methodology for computational trust, [225] have proposed a formal methodology for computational trust development which is limited to the business application domain. [402] has been working on a clearer engineering methodology for building systems that use trust as a component in decision making. Furthermore, [487] proposes a trust engineering methodology which provides design guidance on where and how developers can incorporate trust models into decentralised applications. However, that methodology is too focused on the peer-to-peer application domain and it does not leave room for the multi-disciplinary aspect of trust models and their future extensions. To sum up this section, there are still types of trust evidence and application domains that have not been considered in computational trust engines. For example, the relation between similarity and trust still requires further research [561].

8.2.3 The Problem of Trust Transferability and its Solutions

Self-organising software can be described as a network of autonomous components that cooperate with each other sharing information, and providing and using services. During the life and evolution of such systems the problem of selecting trustworthy components among the ones available is crucial. End-users experience similar problems when needing to select a trustworthy open software of suffciently high-quality. As we mentioned above, computational trust offers a technique to support this decision making process. Techniques such as recommendation systems and social networks could be effective even in open software domains. For example, the reputation of software authors, the feedback shared by users community or rating

values interchanged between autonomic software components are all important factors for identifying reliable services. A basic requirement of such systems is the ability of its components to effectively communicate with others, receiving and sending evaluation's messages that can be interpreted correctly. Unfortunately, in open systems it is not possible to postulate a common agreement about the representation of a rating, its semantic meaning, the cognitive and computational mechanisms behind a rating formation. It may be the case that agents (end-users or autonomic software components) may have different evaluation metric for software quality. Rating software that can be a function of many factors such as programming language used, clarity of the comments, number of bugs reported, tests performed (and by who), number of versions, efficacy and so forth. Differences in evaluation method may invalidate ratings shared by the agents' community. Therefore, the problem is to understand if the parties involved in the exchange of information can actually be considered compatible. In this section we analyse this central problem that affects all the trust-based systems based on the sharing of information. The analysis of this crucial and well-studied problem and its proposed solution is a useful and comprehensive way to describe in depth a generic trust solution. We propose a practical study case on an eBay-like scenario, leaving in the exercise section the study of a parallel situation within an open-software scenario. We will see how the need for a common evaluation language and a correct translation mechanism is the key for making effective use of such trust systems.

8.2.3.1 Trust is Subjective, but Worth to be Shared

Before asking how we can effectively transfer trust, a first question is whether trust has a degree of objectivity. Studies in social science seem to agree about the subjective nature of trust. In the classical definition by Gambetta [195], this is represented by the subjective probability that the trustor assigns to the trustee, that varies according to the trustee, the situation and the level of perceived risk. Any attempt at objective measurement can dangerously mislead agents into thinking that the value is transferable and be used by another trustor, which is not true for trust. In other words, trust is not transitive, which has also been formally shown in [112]. As Luhmann wrote [338]:

> Trust is not transferable to other objects or to other people who trust.

To say that one trusts another without further qualification of that statement is meaningless. But, on the contrary, the success and the diffusion of systems like Social Networks or Ratings Systems make the problem worth to be investigated. Therefore, the problem is to qualify correctly trust judgements, and build a mechanism to translate values produced by two different systems to allow meaningful communications. Jøsang and Pope [289], investigated the transferability of trust by analysing under which formal condition trust may be considered transitive. Their conditional transitivity construct adds conditions for considering trust values, propagated by

transitivity, more plausible. The concept is present also in Abdul-Rahman and Hailes distributed trust model [17]. The conditional transitivity (simplified) requires that:

- *A* has direct knowledge of *B*
- *B* has direct knowledge of *C*
- *A* has knowledge of *B* as a recommender
- *A* can use *B*'s trust value in *C*

Using Jøsang words: *a transitive trust path therefore stops ... when there are no more outgoing referral trust*, where referral trust is the trust in an agent as a recommender. These works clearly show that trust transferability is not a valid concept, but a plausible one that deserves to be investigated. Still, however, transferred trust values can be useful in the decision making process if additional conditions are properly respected and appropriate semantic annotations are added.

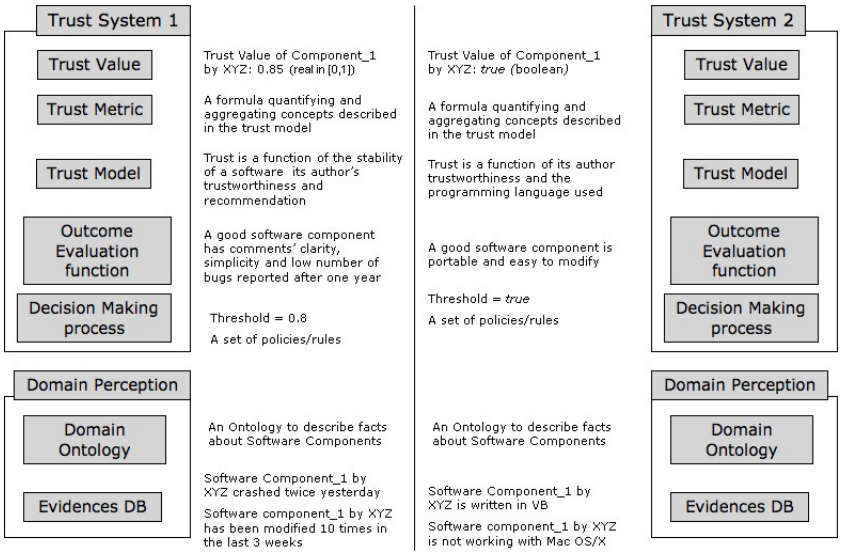

Fig. 8.2 Trust interoperability problem scenario

8.2.3.2 A Generic Model of a Trust-Based System

In Fig. 8.2 we provide a high-level view of a trust interoperability problem scenario where agents have to select trustworthy open software components. The scenario involves two trust-based reasoning systems with parts arranged at conceptual levels based on their primary functionality and purpose. At each level, possible differences between the respective parts of the two systems may cause lack of trust transferability.

Each system is depicted as a multilayer pile with the following components:

- Domain Perception. It includes all information an agent has acquired about its domain of interaction. It generally consists of a domain ontology and a facts database (or evidence DB).
  - Domain Ontology: agent's representation and understanding of the domain it is interacting in. We model it as an ontology that describes terminology and relationships related to the domain under analysis (the open software domain in the scenario under discussion).
  - Evidence DB: the facts collected based on the agent experience which are used to ground trust-based decisions. For example, a factual representation of a piece of evidence can have the form: *The component 1 by developer XYZ crashed twice in the last two weeks*.
- Trust System. It contains the agent notion of trust (referred to as "trust ontology" or "trust model"), its computational model representation by a trust metric, a trust value, a decision making process and a satisfaction function.
  - Trust Value: the quantification of the trust judgement.
  - Trust Ontology (model of trust): the definition of trust in the application context. It defines the elements composing the notion of trust for the particular domain. We can represent a trust model as an ontology. An example description of such a trust model for open software component selection could be: trust is a combination of software component stability, programming techniques used and programmer experience or, in absence of past evidence, the overall recommendation received from other users. The trust metric specifies how these elements are converted into numerical values and how each of them concurs to the final aggregated trust value.
  - Trust Metric: the actual computation performed over the available inputs to generate a trust value. Each concept described in the trust ontology has a computational representation in the trust metric, which quantifies and aggregates all concepts in the trust ontology to produce an overall trust value.
  - Satisfaction Function: a mechanism used by trustor agents to evaluate the quality of interactions carried out with trustee agents. This mechanism can be modelled with a function such as the following:

$$S : O \rightarrow [0,1] \tag{8.1}$$

    Function S maps the set O of possible outcomes to $[0,1]$ assigning a numerical value representing the agent's satisfaction level to each outcome. A satisfaction function models an essential concept in trust computation: the trustor should have a mechanism to understand if the trustee fulfilled its expectations. Using function S, a trustee's trust value can be automatically updated according to the trustor's level of satisfaction. For example, an agent may be satisfied if a piece of software code has clear comments and it does not crash in the first two months. Therefore, the agent can decide to assign to that piece of software a high trust rating.

– Decision Making Process: this component, also referred to as "trust management", describes how the agent exploits the computed trust values to support its decision making process. Generally, based on the agent status and the situation context a threshold T is defined, which specifies the minimum trust value required to start an interaction. The decision making process can be further articulated, and is commonly represented by a set of policies (see the SECURE policy language [82] for instance) or a set of (fuzzy) rules as is the case in the REGRET trust model proposed by Sabater [288].

8.2.3.3 Source of Trust Differences

In any conceptual level of the trust model described above, differences between the respective parts of trust-based systems may reduce the transferability of trust. These differences include:

- *Trust value differences*. Trust-based systems may have different trust value representations. For example, a system may represent trust values with real numbers in the range $[0..1]$, while another may classify trust at discrete levels represented with integer numbers, for example with numbers ranging from 0 to 4 for the case of five trust levels. An additional problem in this respect is the identification of the owner of each representation. This is generally not a trivial problem since different systems may use different names or labels for the same concept or entity respectively.
- *Trust metric differences*. Even when two entities utilise the same trust representation, that is they have adopted the same trust terminology, they may use different trust computation methods. For example, an entity may base 90% of its calculated trust value on the number of bugs reported (for instance assigning it with a weight of 0.9 in a weighted average calculation), while another one may consider number of bugs as a trivial trust parameter.
- *Trust model differences*. Different systems may have different understandings of what trust is. For example, an agent may perceive trust as a prediction of software quality based on the quality of software previously produced by the same developers (a past experience-based prediction), while another agent may consider other factors as well, such as developer popularity and software stability and percistence.
- *Threshold differences*. An agent A can be more optimistic, for example about utilising new software components, and it may therefore have a lower trust threshold than another agent B which may have stricter cooperation requirements. This implies that an exchanged trust value from agent A to agent B could be sufficient for A to start an interaction, even if that would not be the case for B, making thus hard to transfer trust between these two agents.
- *Satisfaction function differences*. Agents can make different assessments of interactions between each other, for instance because of different goals and expectations. For example, in the open software component domain an agent may assess component quality based only on non faulty operation, without taking clarity of

code and number of comments into account. In contrast, another agent may consider confusing source code and lack of comments as significant negative factors affecting trust assessment.

- *Domain representation differences*. Different entities may have different knowledge of the domain structure in which they interact. This can be expressed as differences in domain concept representation, in definition of domain element relationships and in representation of domain element aggregations, resulting thus in differences in evidence descriptions in the domain ontologies of system entities.
- *Evidence database differences*. System entities maintain evidence databases which are dynamically evolvable based on entity interactions, knowledge and recommendations received. In a distributed scenario this is a common situation: each entity has a limited set of experiences and a partial vision of the world. In such cases entities judge trust differently based on different evidence sets and previous experiences.

8.2.3.4 Designing a Solution: Mechanisms to Exchange Trust Values

Without loosing generality, we can assume that a solution to the trust interoperability problem would require entities to partially disclose their trust models to enable trust similarity comparisons between different systems. The degree of similarity produced as a result of such comparisons can be used to weight exchange of trust judgements. For example, if the compared trust-based systems differ significantly, the transmitted trust values can be ignored.

Trust Value Translation

To enable comparisons, mechanisms for translating trust values to a language understandable by the entities receiving trust recommendations are needed. Referring again to Fig. 8.1, a comparison between two systems can be performed at three levels, the trust value level, the trust model level, and the evidence level:

Trust Value Matching. At the level of trust value representation, translations are performed to convert trust values to a common representation. An important aspect of such translations is the estimation of information loss during the trust value conversion process. Pinyol et al. [421] classify trust value representations in four common categories:

1. *Boolean Representation*, where trust values belong to the set $\{0,1\}$
2. *Bounded Real Representation*, typically a real number in the interval $[0,1]$
3. *Discrete Representation*, where the trust value belongs to a set containing discrete elements, such as $\{VeryBad, Bad, Neutral, Good, VeryGood\}$
4. *Probabilistic Representation*, where trust values are modelled using discrete or continuous probability distributions instead of only crisp numerical values.

Pinyol et al. further proposed a method to convert values between these representations which takes into account the uncertainty involved in the conversion outcomes. For example, to convert a boolean representation to a bounded real representation a boolean value of "false" could corespond to any real number less than or equal to 0.5 while a boolean value of "true" could be represented as any value greater than 0.5. An entity using a real number representation would not be able to accurately treat the converted value, and this would represent the uncertainty involved in the conversion process. Pinyol et al. propose to estimate this uncertainty based on the entropy of trust values when considered as random variables. Generally, conversions to richer trust value representations incur higher levels of uncertainty, while no uncertainty is associated with the opposite conversions.

Trust Model Matching. El Messery [375] proposes to enhance trust values by declaring the expectations a trustee has. In this way if two agents share the same expectations it is likely that they will judge situations in similar ways, and consequently trust values can be transferred. The proposed approach requires that trust systems of both agents be able to understand the terminology and the semantics of each trust model. A radical solution implies the matching of the two ontologies describing the trust model, a problem that has not been so far investigated in trust studies. Another possible solution is the definition of a generic ontology for trust representation which will be the starting point for application-specific trust ontologies.

In trust systems research the trust model matching problem has so far been studied using high-level concepts which are compared in taking trust-based decisions. The European project eRep [12] defines a set of terms to represent reputation concepts. The aim of that effort was to define a reputation ontology that all project partners can use as a consensual starting point. That ontology describes in detail all the elements participating in social evaluations, as well as the processes of transmitting them. It also defines the main decisions concerning reputation that agents may take. That ontology is based on the cognitive theory of reputation defined in the book by Conte and Paolucci [122]. Furthermore, Punyol and al. propose a common ontology as a possible common base for mapping different trust models. They describe a generic belief about an entity as a combination of SimpleBelief, a belief that a holding agent acknowledges as true, and MetaBelief, a belief about others' belief.

Finally, an interesting problem concerning trust management in open software is the definition of a common ontology for describing software quality, which generally has the form of a common dictionary for expressing software component ratings. Elements of that ontology may include various software quality parameters such software stability (represented by variables such as number of crashes, and numbers of modifications, patches and versions), age, clarity of comments, complexity of the code, reliability of the developing language used, portability, reputation of the authors, tests passed and reliability of the testers.

Evidence Matching and Unsupervised Similarity Learning. If agents act in the same or similar domains, it is likely to encounter similar situations. A degree of sim-

ilarity can be deduced by simply matching evidence and their corresponding trust values. This could be performed in several ways, but the common idea is that interacting agents disclose sufficient information to compute a degree of compatibility based on common data or behaviours in specific situations. This class of solutions does not consider the elements of the trust model, but instead it focuses on the comparison of common data aiming to understand the similarity between different trust systems. Therefore, this approach limits the need for a common ontology. However, it still requires that interacting agents communicate using a commonly understandable language.

A representative example of this solution approach is the collaborative filtering paradigm. Collaborative filtering assumes that ratings originating from agents with similar preferences are the most accurate. The main aspects of collaborative filtering mechanisms concern storing and exchange of user profile and recommendations. Upon entering the system new users provide information about their profile and a number of trust recommendations. All new user input is stored in a central database. Subsequently, when a user requests for a recommendation, the system will compare the requesting user's profile with those stored in the database, and it will suggest trust recommendations from similar users. The efficiency of such systems depends on the number of recommendations they receive, the number of users in the system and the level of detail of each user profile. Collaborative filtering systems are typically centralised, commonly based on a central database that stores user profiles and past user recommendations, and therefore they are not always a feasible solution.

Alternative solutions usually extend the basic idea of collaborative filtering, namely searching for similarities among user related information, in a dynamic and distributed fashion. For example Wang in [527] proposes a recommendation system where idle agents "gossip" periodically with each other, exchanging information about their assessment of interactions they had previously held with other agents. Interaction assessment is commonly done using probability functions which reflect agent opinions about interaction participants. The result is a similarity factor in the range $[0, 1]$ used to weight agent recommendations.

Trust-based System Comparisons

In general, automatic comparison and matching of trust-based systems can be performed based on the satisfaction function S, the trust value T and the evidence database DB. In all cases the aim is to provide each agent with an estimate of either the trust metric T or the satisfaction function S of the agents with which it interacts. T and S are generally, but not always, correlated. In particular, the values of T are commonly calculated based on past values of S, except in rare cases, for instance when no information about relevant past interactions is available. Therefore, knowing T does not generally guarantee the expected S. For example, the fact that a person has been reliable in interactions with a specific person does not guarantee the same reliability in interactions with other persons. Similarly, a high satisfaction

value *S* does not generally guarantee the quality of a recommended trust value, since in the computation of *T* the role of *S* is unknown.

Trust comparison approaches are generally based on sharing trust values, on sharing past interaction evidence, on directly comparing satisfaction function *S* and trust metric *T* values using an agreed common domain ontology, and on approximating function *S* and metric *T* values based on stereotype scenarios. More specifically:

- *Sharing Trust Values*: In this approach agents share trust values concerning other agents which they have calculated previously. The idea is that interacting agents check for common acquaintances and they use the respective trust values to to compute a compatibility degree, which they subsequently use to weight trust-based decisions. In calculating compatibility, statistical indicators such as correlation can be used, and supplemental information such as number of accepted/rejected interactions with an agent can make the computation more plausible. Furthermore, it is assummed that agents have universally known IDs, although this hypothesis is not always valid. In addition, if agents have different trust value representations, a conversion may be performed, for example using the method described in [421]. That method predicts trust metric *T* without requiring any knowledge of the agent trust model, and for sufficient number of acquaintance agents it can be a quite accurate indicator. However, it suffers from poor privacy and communication overload due to the high amount of information exchanged.
- *Sharing Interaction Evidence*: This approach is also based on information sharing but here sharing concerns assessments of single interactions instead of overall trust values. In other words, the goal is to predict function *S* instead of metric *T*. In this approach, communication overload is even higher, but no significant privacy issues are involved since the exchanged information concern scenario snapshots instead of agent private data.
- *Direct Comparison of Function S*: When there is a common ontology describing system facts, each agent can easily map its function *S* over that common ontology and directly compare it with others. The simplest case is when the function *S* of all interacting agents has the same basic form (for example a linear combination of factors). An example of such an ontology is the recent evolution of the eBay feedback system (see Fig. 8.3), where four fixed criteria have been introduced to assess the validity of an item sold, representing a first common base for comparing feedback. By directly comparing the two functions, the agents compute an accurate degree of compatibility, without disclosing sensible information about other agents or personal experience, and with low communication overload unlike the previous two approaches.
- *Predicting S and T using Stereotype Situations*: When there is no common ontology, but the agents at least partially can understand each other, a solution can be build by using stereotype situations. Here we describe the prediction of function *S*, but the method can be applied to the prediction of trust metric *T* using trust values instead of values of satisfaction, and stereotype agents instead of stereotype situations. The aim of this approach is to accurately predict the function *S* using the least number of messages. In the general case function *S* is assumed to

be defined from a set of domain concepts, represented by multiple variables, to a value.

$$S1 : f(X1, X2, ..., Xn) \tag{8.2}$$

$$S2 : f(Y1, Y2, ..., Ym) \tag{8.3}$$

We assume that, if agents use different value representations, they translate them using the technique described in [421]. Agents send stereotype situations, which they consider meaningful, to other agents and waits for them to evaluate the the situation. For instance, an agent considering the low shipping time essential for being a good eBay seller may propose two situations where this factor varies drastically. Agents can propose situations where only one key-factor changes, in order to understand the importance of that specific factor, with the drawback of not understanding the mutual dependence of the factors in the formula. In general, agents need a strategy to generate the appropriate next situation after having received feedback from other agents. The strategy should indicate when the process should stop, that is when enough information has been collected to understand other agent model. In general, agents may employ an unsupervised learning system or adopt statistical tools such as regression and correlation to understand the reasoning model of other agents, performing an on-the-fly negotiation of their preferences. This solution appears to be a good trade-off over the previous ones: using stereotype situations sensible data are not disclosed, communication overload is relatively small, varying from the perfect situation of the evidence sharing approach to the case where a large number of messages will have to be exchanged to understand other agents. The required number of messages will generally depend on how close the interacting agent representations are and on the number of situations proposed that are relevant and fully understood.

8.2.3.5 Privacy Issues

The disclosure of extra information instead of a trust value raises an issue of privacy. Interacting parties may not want to share information about their trust judgements or reasoning models, even in open software evaluation. The problem was described by Seigneur [468] as the trade-off between trust and security:

> There is an inherent conflict between trust and privacy: the more knowledge a first entity knows about a second entity, the more accurate should be the trustworthiness assessment; the more knowledge is known about this second entity, the less privacy is left to this entity. ...A solution should allow the benefit of adjunct trust when entities interact without too much privacy loss.

Referring to the proposed solution, it becomes evident that a matching performed at trust model level discloses less information than the explicit sharing of evidence and the comparison of trust values of common acquaintances. Several systems have

been implemented to add an extra-layer of security based on trusted computing and encryption key policies to guarantee the confidentiality of the shared information. A recent work done by Cissee and Albayrak [114] described an approach based on multi-agent systems for preserving privacy in information systems, where all communication takes place in a trusted environment and a third entity is elected as referee, collecting all required encrypted information, performing the matching and sending the results to the entity involved.

8.2.3.6 Practical Case Study: Enhancing the eBay Feedback System

In this section we briefly describe the eBay feedback system, whose evolution represents an excellent case-study for the above discussion.

eBay provides a feedback system aiming to provide buyers with an additional decision support tool for selecting the most reliable sellers. Introduced in 1998, feedback systems evolved adding interesting features from the perspective of the above discussion. The eBay feedback system has a boolean representation for each feedback, where +1 is the value of a positive feedback and -1 of a negative one. The feedback score is simply computed by summarising all feedback messages received. In addition, a buyer may add a comment in natural language.

eBay has introduced several new features to support a better decision-making process and add more semantic value to the original unqualified value. Fig. 8.3 depicts the information displayed to a single user. Since March 2007, eBay introduced detailed ratings, where users rate the buyer according to four categories: item as described, communication, dispatch time, postage and packaging charges. These subcategories disclose more information to support a more accurate trust value semantic, resulting in a type of common ontology for representing buyers/sellers ratings. For each criterion a discrete level feedback is proposed with 5 possible levels ranging from very accurate to very inaccurate. Other interesting information used to define trust values includes the number of ratings of each category, the temporal distribution of the feedback, the number of feedback messages, the feedback score of the rating entities and the number of feedback removed.

We see the eBay feedback system evolution as an answer to the problem of unqualified ratings, a way for defining a common ontology based on which different ratings can have a clearer and more objective meaning for the buyers. It is therefore a good template that many other recommendation-based systems, including the ones used for self-organising software, could refer to.

Detailed Seller Ratings (since May 2007)

Criteria	Average rating	Number of ratings
Item as described	★★★★★	52
Communication	★★★★★	51
Shipping time	★★★★★	52
Shipping and handling charges	★★★★★	52

Fig. 8.3 Ebay feedback system

8.3 Applications

This section applies the above approach to trust computation and transferability in two application domains: the selection of open source software components, for example, to be used at the time of software self-organisation, and in development of open collaborative software such as Wikipedia.

8.3.1 Selection of Open Source Software Pieces based on their Trustworthiness and Reputation

A major asset of OSS projects derives from their collaborative and community values [484]. On one hand it has been argued that the more OSS actors participate, the higher software quality is reached. In this respect Raymond argued that "...given enough eyeballs, all bugs are shallow..." [440]. On the other hand, recent investigations have argued that few actors review OSS code and that this practice may not be enough to sustain high quality in OSS projects: "...the vast majority of 'eyeballs' apparently do not exist..." [460]. As the collaborative and community values are deeply anchored in the OSS ecosystem [484], the root of the problem is related to a lack of enabling technical mechanisms. The OSS community is present and willing to contribute but the tools needed to contribute are missing or too cumbersome to use. A few repositories aggregating OSS datasets and dashboards, such as Web portals presenting the information, have recently emerged but there is a lack of automated exchange and interoperability: the first workshop trying to harmonise them was run in June 2006 [224].

Fortunately, with the advances in Web technologies, often termed Web 2.0, it becomes increasingly easier to build tools for networking the OSS community. The EU-funded EDOS project [467] has built a formal information model, termed the Project Management Interface (PMI), [412] which describes OSS artefacts, such as Actor and Platform or Maintainer, and OSS activities for example SubmitPatch activity and SubmitTestReport activity. In EDOS, a number of tools have been deployed to gather and distribute information about OSS projects in a peer-to-peer manner, by reusing Web 2.0 building blocks such as XML, RSS feeds and REST/Web services [179]. For example, a Quality Assessment (QA) Web portal, termed the EDOS dashboard, has been put in place to easily inform OSS actors of the quality of OSS projects. The quality of these projects is estimated based on information reported by other OSS actors who can deploy user-friendly tools on their platform when needed to easily evaluate specific OSS projects. Therefore, such actors and their platforms play an important role in the EDOS-powered community process with collaborative OSS quality assessment and improvement. However, relying on external information submitted from an open ecosystem, decentralised by nature and populated by possibly competing actors, introduces a number of trust issues, for example competing actors, including actors outside the OSS community,

may try to submit false reports about the quality of projects of their competitors. Furthermore, cheating is facilitated when actors can enter and leave the system without any control from a centralised authentication service; and in the QA application domain, "trust in the accuracy of any test data ...depends on ...trust in the providence of the testers" [172]. It seems fair to assume that in small projects all OSS QA team actors may know face-to-face all the involved testers. However, this may not be possible in larger projects because the OSS ecosystem is an open environment where actors and their digital identities can come and go. Nevertheless, due to the need of more actors carrying out test tasks, as long as testers are trustworthy, even not personally known testers are welcome to contribute to the project quality improvement process. It is especially relevant in the EDOS project that relies on a peer-to-peer storage layer for OSS information. There is the need for security/trust metrics to select the most trustworthy peers for efficient and safe distribution of QA information.

If we assume that there are is a sufficient number of trustworthy testers, tests and platforms, the correlation of all test reports and defects should allow us to detect untrustworthy tests and platforms. For example, tests that always fail despite that the software as such would work or compromised platforms sending false defect reports. More precisely, if one out of fifty platforms with the same hardware and software configuration reports that numerous different tests crashed the platform whereas all other platforms report that all these tests passed, there is a chance that the platform itself is buggy and therefore it should be considered as untrustworthy. Another reason for deviant reports may be that testers are untrustworthy, for example, because they are involved in competing projects or lack necessary testing skills.

Recent work on test quality has emphasised the importance of tester quality: "your trust in the accuracy of any test data will depend on your trust in the providence of the testers/quality of the testers" [172]. Although that related work is a first step towards taking the importance of tester quality into account, it uses only fixed, manually configured, subjective trust levels to characterise tester quality. An example is the the average testing quality which is estimated by the QA manager. The EDOS QA information model and approach allow us to easily collect and distribute tester quality and trust objective evidence. There are different aspects that can be considered regarding tester quality and trustworthiness. To start with, there are differences between end-user actors that are only willing to run specific tests on their platforms and tester actors who have designed tests in the first place. Secondly, a tester should be rewarded when a test carried out before the software release detects bugs, especially critical bugs. Furthermore, if the released software contains many critical bugs, the testing quality should be questioned. Since there are chances that testers make mistakes on purpose due to their participation in competing projects, we use the term tester "trustworthiness" instead of tester "quality". Fig. 8.4 depicts our initial decomposition of the testing trust contexts of testers. We assume that each rectangular trust contexts in Fig. 8.4 is associated with a counter counting the number of times a tester has achieved a positive outcome and a negative outcome in this trust context. We use the term "test report trustworthiness" to refer to the number of

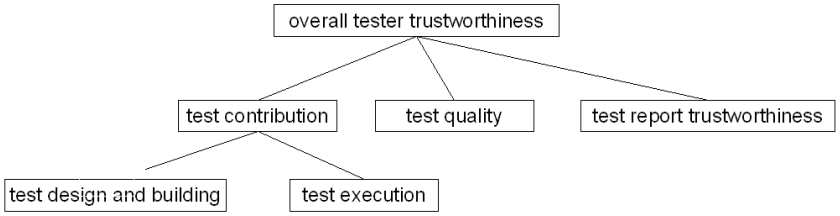

Fig. 8.4 Example of the Trust Contexts of a Tester

times a tester has reported the same test outcome with the majority of testers with similar platform configuration. The context test contribution captures the number of times the tester has contributed and spent time for the OSS community on QA tasks. The activity diagram below depicts an approach to to update these counters. Depending on the criticality of the bug found by an end-user actor when using the

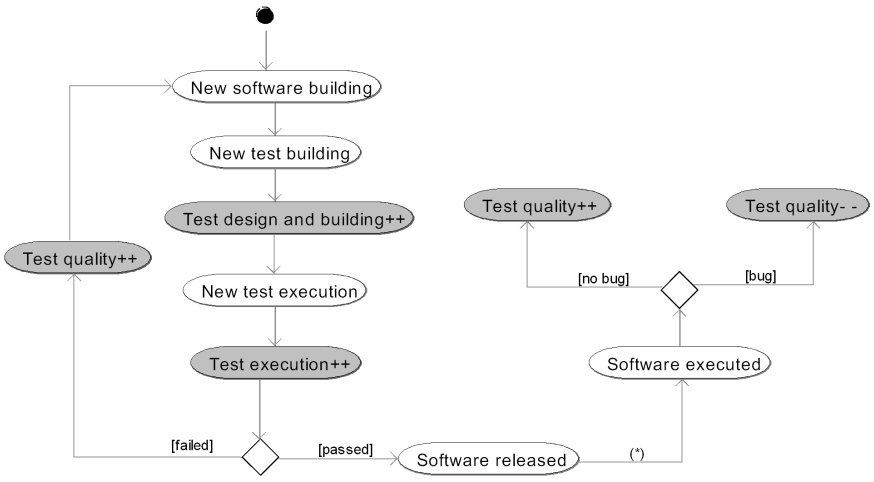

Fig. 8.5 Example Update of Tester Trust

software, we can modulate the trustworthiness counters of test quality based on bug criticality. For example, given:

- a tester u;
- N independent tests carried out by u;
- $crit$, the criticality of a bug found to be linked to a specific test that ranges within [0,1]

an estimation of test quality trustworthiness of tester u, denoted by $tqt(u)$, can have the form:

$$tqt(u) = \frac{Sum\ of\ (1 - crit)\ for\ all\ tests\ carried\ out\ by\ u}{N} \tag{8.4}$$

8.3.2 Trust in Open Collaborative Authoring Applications such as Wikipedia

The emerging self-organising nature of the Web 2.0 shows similarities with the production process of an open-software component. With the proliferation of new tools such as wikis and blogs, that simplify and democratise publications, the Internet appears today as the collaborative work result of an open community. The Wikipedia project, under analysis in this section, represents one of the most successful and discussed examples of this trend, with more than 1.2 million registered users (English version) and more than 2 million articles [2]. In a smaller scale, a self-organising open software is also the product resulting from cooperation of a community of developers, where the code added by each member represents a contribution. Therefore, it is not absurd to presume that the two domains may partially share problems and solutions. In this section, we analyse how computational trust techniques have been applied to wiki-based applications to provide users with a tool for supporting more effective decisions. We describe an autonomic trust model that was originally designed for the Wikipedia project, and has lately been generalised for collaborative self-organising information systems. We envisage the possible application of such a computation in open collaborative software.

8.3.2.1 Wikipedia internal quality mechanism

The Wikipedia project represents the most successful and discussed self-organising information system. It is regarded as one of the best examples of collective knowledge, a concept that is often lauded as the next step towards truth in on line media. The problem of article trustworthiness is central to Wikipedia future development as a recent extraordinary case has brought to attention. Nevertheless, the growing interest and utilisation of such types of source remains unquestionable. In 2006, a study by the magazine Nature showed how, for a subset of scientific-related articles, Wikipedia was actually of comparable quality than Encyclopedia Britannica, with an average of 4 inaccuracies in Wikipedia against 3 in Britannica [211]. To address this concern, Wikipedia has defined an internal rating mechanism to classify article quality. The mechanism is an articulated centralised recommendation system, where any decision is taken based on collected ratings and opinions of Wikipedia users. Using this system each Wikipedia article may receive a quality classification or a review. In particular, therer are two highest quality article classification levels,

The first one concerns is the "featured article" status, which means that an article has been identified as one of the best articles produced by the Wikipedia community, and it is particularly well-written and complete. Only 0.1% of articles are featured articles. The second one is the "good article" status: the article contains excellent content but it is unlikely to become featured in its current state. For example it may be too short, or involving a too specific topic, or a topic on which there is a shortage of information.

The Wikipedia internal mechanism is a robust tool that proves how an efficient recommendation-based approach can increase the quality of the information shared. Anyway, it is not the ultimate tool and it is not free from drawbacks. The rating system can be slow to react in comparison to the fast-changing nature of Wikipedia, making its suggestions out-of-date. New strategies could be coupled to this approach. An interesting Wikipedia feature makes it possible to define complementary strategies to compute articles trustworthiness. Wikipedia has been designed so that it keeps a completely transparent database of all the past contributions. The history of each page is accessible, providing information regarding authors and differences with previous versions. The accessibility of this information provides a valuable base that, if properly processed, can ground a solid decision making process, as shown in the next section.

8.3.2.2 Alternative Approach: Computing Trust using Domain Elements, from Heuristics to Generic Trust Patterns

Wikipedia has been the target of computational trust experiments only recently. A common factor of such efforts was the idea of defining strategies for trust value computation further to the recommendation approach. The common idea was to extract evidence of trust presence directly from application elements. This presumes that the majority of trust information is represented, even implicitly, in elements and dynamics of the particular application. For trust computation to be effective, the identification of these elements and their trust meaning should be plausible and justified in the particular application context. This approach to trust computation is a subclass of evidence-based trust where the evidence used are internal application elements or dynamics. Such computation results are application-contained and thus non invasive, not requiring any infrastructure added to the application for assessing trust, unlike recommendation systems.

A first limited experimentation was performed by McGuiness [369]. To assess the trustworthiness of a Wikipedia article, the author applied heuristics based on a version of the famous PageRank citation-based algorithm. Many authors, notably Massa [359], identified in Google PageRank all the elements of an application-contained trust metric. In that specific case, the application is the whole Web, seen as an interconnection of mutually linked web sites. PageRank considers the outgoing and ingoing links of a web page as trust evidence. In Wikipedia, the authors considered the relative number of times the name of an articleappears as a link in the whole encyclopaedia. The formula proposed is the following:

$$Trust\ doc(d) = \frac{occurrences[[d]]}{(occurrences([[d]]) + occurrences(d))} \tag{8.5}$$

where d is an occurrence of an article name as a plain text and $[[d]]$ an occurrence of the article name as a link. In other words, in that approach the fact of being a link was selected as evidence for the trustworthiness of an article. However, its applicability in that context may be severally argued: an expert Wikipedia user may argue that in Wikipedia there are automatic procedures that link articles, or that an author may link articles independently from their quality, for example for the sake of completeness. This example, similar to many others, shows several concerns regarding how to perform a selection of evidence directly from domain elements: the selected heuristics cannot be directly applied without critical analysis of their plausibility and trust-related meaning in the application context. The McGuiness experiment does not go beyond definition of ad-hoc heuristics, lacking objectivity and systematicity. Its lack of plausibility for expert-users decreases drastically the meaning of trust computation. In the next section we will describe a trust model developed to address these issues.

8.3.2.3 DANTE: a Trust Model for Collaborative Self-Organising Information Systems

An open and collaborative self-organising system follows an evolution from start-up to a mature status, where the system auto-corrects its errors and changes in response to external stimuli in order to survive and continue to provide its service. During this evolution, some internal dynamics and elements emerging from system interactions reveal information about the health status of the system. This status information is an indicator of the ability of the system to cooperate as expected, its reliability and the probability for it to fulfill other expectations. Therefore the correct identification and quantification of this information is a valuable and relevant source about system's trustworthiness. This is the idea that grounds the trust model for collaborative self-organising information systems described in this section. The trust model, termed DANTE (Domain Analysis and Trust Extraction) [158], defines a set of trust factors which are indicators of the health status of the system. Their presence is evidence of system reliability and trustworthiness, while their absence is a warning for the agent taking a trust-based decision. That trust model was originally designed for an experiment in the context of the Wikipedia project, but it was later generalised by the authors to cover any collaborative self-organising information system emerging from the collaboration of different authors. Anyway, the model is not regarded as a completely domain-independent solution.

The DANTE model consists of a set of guidelines that can be used to design a trust-based solution, where domain-specific expertise plays a limited supportive role. Each trust factor can be described as a generic trust pattern which elements of the application can match. The selection of an element is therefore justified by the corresponded trust factor. Trust factors are grounded on social science theory

of trust which guarantees that the computation performed has a meaning for trust, while the fact that the mechanisms are generic provides a valid basis for keeping the computation less domain dependent. Trust factors should therefore limit the space for unsystematic and unjustified heuristics.

Among the benefits of this approach lies the fact that it does not require any additional explicit data to be added to the application and, by performing computations over system elements, it is a feasible non-invasive solution. Moreover, that approach can be used to enhance the decision making process of an autonomic system.

It is essential to specify the meaning of trust factors. Trust factors are not intended as definitive evidence of system trustworthiness, and their absence is not intended as a definitive proof of system unreliability. On the contrary, trust factors are considered as plausible trust indications. The core of the computation of the proposed model aims to quantify the plausibility of the selected system elements and the uncertainty of the respective conclusions. Each factor used requires a number of critical questions assessing its plausibility when applied in a specific context. These critical questions require an investigation of the specific situation context to be answered, and they are not dependent on the overall trust domain.

8.3.2.4 Trust Factors for a Self-Organising System

The trust factors identified are divided in the following categories: pluralism, temporal factors, stability, activity degree, similarity and categorisation. Each category interacts with the others to strengthen or weaken their conclusions. In the rest of this section we further describe each trust factor while in the next section we show how they were applied in the context of the Wikipedia project.

- *Pluralism*: A collaborative self-organising system is characterised by contributions of many actors whose input affects system dynamics. According to the fundamental trust pluralism principle, in a purely open collaborative environment there should be an appropriate balance between actor contributions. In particular, the cases where the system depends on contributions from only too few or too many actors should be avoided. The former case has the disadvantage that the resulting system can be highly biased, while the latter can lead to a fragmented system with increased complexity and low coherence and consistency among its parts. This is clearly demonstrated in the collaborative document authoring approach that is followed in Wikipedia. In the Wikipedia approach the authoring status of an article can range from having been edited by only one or a very limited number of contributors leading to a lack of pluralism, to being the collaborative authoring result of a large number of authors, each providing a small contribution, resulting in highly fragmented information. Fragmentation may lead to inconsistent editing or even conflicting ideas, while lack of pluralism may lead to biased information. Therefore, an article is considered as highly trustworthy only if it has been edited by a sufficient number of authors, who have each contributed significantly resulting in high article pluralism. Of course, it may be the case that an article written by a single author be of exceptionally high

quality, but this case is less plausible and more risky because if that single author is fallacious, the article will not be trustworthy. On the other hand, if an article has sufficient pluralism, it is more plausible that a solid checking mechanism has been used.

For pluralism to be properly assessed, contributions should come from recognisable authors. More specifically, it is only required to distringuish between different contributors and not necessarily characterise them based on their identity or trustworthiness. Furthermore, the pluralism principle is more plausible when the total number n of contributors is relatively high, the contributors can be considered as independed or not explicitly related, and the number m of contributors with a non-negligible contribution is significant. Finally, contributions should occur with a relatively high frequency to enable for sufficient contributions to be made in any period of observation.

- *Stability*: Self-organising systems take auto-corrective actions to improve their offered services when they perceive themselves as not exhibiting the behaviour intended for their surrounding environment. During such state transitions system components may reach states where their integration and functionality had not previously adequately evaluated. On the other hand, self-organising systems that have reached a high degree of maturity have less reasons to radically change their core behaviour apart from perhaps adding some new functionalities. Therefore, they are considered as stable. Stability implies that system components are well-established and reliable, and that their evolution has reached a mature stage. In contrast, instability indicates the existence of components that still need to evolve to correct their behaviour and abilities.

  The role of stability in trust evaluation has been widely examined in the literature. Frewer and Miles in [193] show that temporal stability is directly linked to perceived trustworthiness. Different bodies (public and private) were asked to release the same information concerning a number of food hazards. The sample of people involved in the test tended to consider the provided information more trustworthy if they had been released by a body with higher temporal stability. For example, hospitals had a higher trustworthiness characterisation than governmental bodies. The Standford Persuasive Lab Guidelines [1] attributes to the permanence and stability of the information on the Web one of the main five sources of credibility and trust. In the context of Wikipedia, an article stability is considered as evidence that the article contains information with a high degree of completeness and acceptance, without being the source of contentions and editing revisions.
- *Activity*: This trust factor concerns the use of system activity as an indicator of system trustworthiness. In this view, inactive systems should be trusted only after detailed further investigations. In contrast, the trustworthiness of active systems should be considered as plausible and its degree should be further analysed using appropriate plausibility tests. Examples of parameters that should be included in such plausibility tests include the quality of system activities, the number of competitors providing similar system services, and the number of requests for each provided system service. In Wikipedia, the numbers of edits and visits received

were considered as article activity indicators. In particular, it was considered that stable articles with high numbers of edits and visits received could be characterised as trustworthy. This assertion can be further strengthened by considering article importance. In Wikipedia, the number of links pointing to an article were used as an indicator of its importance and that value was further used to normalise the expected activity of that article. For example, based on the number of existing links an article such as "Garda Lake" would be expected to demonstrate less activity than an article such as "USA".

- *Temporal Factors*: Temporal factors [334] are deduced exclusively by processing the time distribution of the activity of the actors/components that participate in a self-organising system. Therefore, a number of them partially overlaps with the activity and stability factors described above. Temporal factors are defined as follows:
  - *Regularity - Persistency*. This factor examines whether system activity is persistent over time. It is considered to have a positive value if for a given time interval p at least one interaction takes place within the system for each interval p over the selected observation period.
  - *Frequency*. The factor checks the average period between two interactions and the variance of this interval.
  - *Presence*. This factor examines the amount of time the system has been in operation, that is the difference between the first and last system activity.

  Temporal factors consider the constant system operation as evidence of the system ability to fulfil user expectations. For example, if a system is relatively new or it has experienced long periods of inactivity, frequent service interruptions, high service provision variance and low interaction frequency, then its trustworthiness would be characterised as low and it would require further investigation. Temporal factors do not examine the overall system stability, but only the temporal stability of system activities. This means that the system may change its properties and functionality as needed, but if the system activity remains stable over time temporal factors contribute positively to system trustworthiness. In Wikipedia, the activity whose temporal factors are taken into account in trustworthiness estimation is the action 'edit' carried out by article authors.
- *Similarity and Categorisation*: This trust factor has a statistical nature with clear human related meaning. The assumption is that entities showing properties significantly different from the average of their category, that is they are 'outliers', should not be granted trust without further investigation. For example, if a system does not comply to standards, this would be an indicator that the system was produced by non-experts in an unsystematic manner and without proper testing and taking state-of-the-art into account.
  Tversky [503] have extensively studied similarity and categorisation as cognitive mechanisms used to make judgements under uncertainty. Furthermore, in the context of computational trust similarity has been investigated by several authors, such as Ziegler and Goldbeck [561], while Castelfranci and Falcone [99] in their cognitive trust model consider categorisation as a first form of trust-based

reasoning. The plausibility of the trust factor increases if the category set has relatively large cardinality and low variance. Moreover, plausibility further increases if the categories used are well defined, the system entities have comparable lifetime in the environment and the population is stable, meaning that it is not in the process of evolving.
In Wikipedia, this approach has been applied based on the Wikipedia article categorisation mechanism. Articles of comparable topics and importance are expected to comply to the standards emerged by the collaborative system dynamics. If an article is not compliant to such a standard, this implies that the article would require extra editing work, or that it was not edited by expert authors aware of Wikipedia emerging trends and guidelines.

Examples of trust factor computation in Wikipedia context are depicted in Table 8.1. To identify the distribution properties of the calculated variables, statistical quantities such as average and standard deviation were used.

Trust factors	Comments
Pluralism	Number of reverted edits. A reverted editing is a roll back to a previous version of an article, rejecting any intermediate modifications
Activity	Number of links to the article
Activity	Number of visits
Activity	Number of edits
Temporal Factors: regularity	Average and standard deviation of the time intervals between consequtive edits
Temporal Factors: persistency	Number of intervals in which there has been at least one interaction
Temporal Factors: presence	% Time between last and first edit
Stability	Percentage of article edits from time t to present time
Stability	Percentage of text differences between article versions at time t and current article versions
Pluralism	Average number of article edits per user
Pluralism	Standard deviation of article edits per user
Pluralism	% of edits produced by the n most active article users
Pluralism	% of edits produced by users with more than n edits for a given article
Pluralism	Number of discussions (talk edit)
Categorisation	Computed by relying on Wikipedia categorisation of articles. Among the elements considered to evaluate similarity: length of the text, images, variance of sections, references.

Table 8.1 Trust Factors in Wikipedia context

8.3.2.5 The Wikipedia Experiment

In this section we present the results of computing the above described trust factors in Wikipedia. The experiment was conducted on 7718 Wikipedia articles. These articles included all 846 featured articles, namely special articles considered by Wikipedia as having the best quality, and the most visited pages having at least 25 edits. These articles represent the 65% of the editing activity of Wikipedia and the high majority of its access, thus they can be considered as a significant set. The results are summarised in Fig. 8.6. The graph represents the article distribution based on their trust values. We have isolated the featured articles (grey line) from standard articles (black line) and the hypothesis was that featured articles should show higher trust values than standard articles. The results obtained clearly showed a trust value difference between featured and standard articles, which had trust values of around 45-50% and 75% respectively. Furthermore, 77.8% of featured articles is distributed in the region having trust values greater than 70%. In addition, 42.3% of standard articles are distributed in the region having trust values less than 50%, which does not include any featured articles. Only 23 standard articles are in the region $> 85\%$, where there are 93 featured ones. The experiment, covering articles from different categories, was conducted on an absolute scale, and it shows a minimal imprecision if compared with a previous experiment conducted on a set of 200 articles, all taken from the same category "nations" [158], where we could rely on relative comparisons of similar articles. This shows that the method has a general validity.

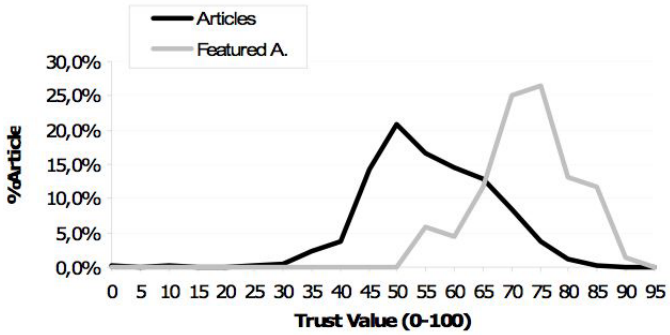

Fig. 8.6 Trust calculation results in Wikipedia context

8.4 Conclusion

There are different computational trust models and metrics. Instead of forcing all the trustors to change their trust model and metric to one yet-to-be-found-and-adopted

trust model, it seems more feasible to provide means for trustors to obtain an understanding of trust models used by others, and to translate the trust values and received recommendations to their own trust model. Finally, another use of trust models in self-organising software is to take trust into account when characterising software testers and developers.

Problems - Exercises

8.1. How would you design a system that would automatically select the most trustworthy open-source software building blocks needed for a particular application?

8.2. Consider an eBay-like decentralised database. Agents have different trust models applied to the available system data, but they have the same domain representation, that is they understand each other regarding eBay related issues. Since the environment is decentralised, agents have knowledge only about their past interactions. In the above context, provide examples of all possible differences between two agent systems. Consider all trust system layers.

8.3. For the scenario context described in Prob. 8.2 consider that each buyer utilises the same domain representation described by the eBay database structure. Morevover, each agent has a different trust model represented by an ontology that is unknown to others. The outcome of an interaction is based on a 4-tuple f_1, f_2, f_3, f_4 representing the four eBay criteria:

- f_1: Item as described
- f_2: Communication
- f_3: Dispatch time
- f_4: Posting and packaging charges

Furthermore, each agent has the following capabilities:

- A trust value representation by a real number in the range $[0,1]$
- A trust metric to produce trust values which has the form $T = a{\cdot}A + (a-1){\cdot}B$ where A is the trust value derived from past interactions that are saved in the agent database and B is the value collected from received recommendations. In absence of interactions, the trust value A is set at the half of the scale (equal to 0.5). After each interaction the value of A is updated using the function S to evaluate the quality of the interaction. The value a is a coefficient that determines the relative importance of received recommendations versus previous agent interactions. If $a = 0$ then only recommendations are taken into account in trust calculation. The value of A after an interaction is given from the formula $A = (b{\cdot}A + (1-b){\cdot}S)$ where b represents the importance of past agent history in comparison to the last interaction. The value b is a coefficient that describes how the trust value changes after a new interaction. If b is close to 1 the impact of new interactions over the stored trust value is negligible, and if b is close to 0 newest interactions strongly affect calculation of new trust values.

- A set of evidence derived from agent past interactions. Evidence can be stored as tuples of the form:
  $buyerID,\ f_1,\ f_2,\ f_3,\ f_4,\ date\_of\_transaction,\ price$
- a database with the trust values for each buyer stored as tuples of the form:
  $buyerID,\ trust\_value,\ number\_of\_interactions$
- A trust threshold T used by agents to drive purchasing decisions based on the respective buyer trust values
- A satisfaction function S, defined as:
  $S(item) = c_1 \cdot f_1 + c_2 \cdot f_2 + c_3 \cdot f_3 + c_4 \cdot f_4$
  where S is a value in the range $[0,1]$

The values a, b, c_n and T are random for each agent to model the differences between trust systems of different agents.
Consider a group of N sellers and M buyers. To decide whether to buy an item from a particular seller, an agent A can use the respective trust value stored in its internal database, if such a value had been previously stored, or ask an agent B for a recommendation. After a purchasing interaction has taken place, the satisfaction function S is used to assess the quality of the purchasing outcome and characterise the trust quality of the respective buyer.
(a) How can a buyer evaluate if it would be preferable to act in isolation or to share the trust values calculated from previous interactions?
(b) Design an automatic strategy to translate trust values from one trust system to another so that the overall quality of interactions is improved. How can this strategy be evaluated?
(c) Can you think of a similar scenario in a self-organising software system? Which trust metric and which satisfaction function S would you suggest to assess software quality? Which are the possible differences between these two metrics? Can you define a common ontology to describe both software quality and trustworthiness?

8.4. Identify critical questions that can be asked regarding the stability trust factor. Do you consider open software stability as plausible evidence of its trustworthiness?

8.5. What is the meaning of the trust factor "pluralism" in a self-organising open-software scenario?

Key Points

- Computational trust and reputation can assist in improving safety and quality of a self-organising software ecosystem;
- there is a need to consider trustworthiness of testers and developpers as well as the approaches followed by trustors to model and estimate trust and reputation.

Further Reading

Reputation Management Services. A book chapter in the "Computer And Information Security Handbook", edited by John Vacca, which goes into details of reputation management services. The other chapters of that book will remind the reader of other security aspects for successful software self-organisation (J.-M. Seigneur, 2009, Elsevier, ISBN:9780131426436.)

Collaborative Computer Security and Trust Management. A book that covers the social engineering aspects of collaborative software (J.-M. Seigneur and A. Slagell, 2009, Information Science Publishing, ISBN:978-1605664156.)