Trust in Social Machines: The Challenges

Kieron O'Hara[1]

Abstract. The World Wide Web has ushered in a new generation of applications constructively linking people and computers to create what have been called 'social machines.' The 'components' of these machines are people and technologies. It has long been recognised that for people to participate in social machines, they have to trust the processes. However, the notions of trust often used tend to be imported from agent-based computing, and may be too formal, objective and selective to describe human trust accurately. This paper applies a theory of human trust to social machines research, and sets out some of the challenges to system designers.

1 INTRODUCTION

Computers have always been sociotechnical systems, embedded in organisations, or serving the purposes of users for work or leisure. However, thanks to the spread of interactive read/write technologies (e.g. wikis, photo-sharing, blogging) and devices and sensors embedded in both physical and digital worlds (e.g. GPS-enabled hand-held devices), people and machines have become increasingly integrated. Terms such as 'augmented reality' and 'mediated reality' are in common use, and the embedding of computation into society via personal devices has led to discussion of *social machines* and *social computation*, an abstract conception in which people and machines interact for problem-solving. The 'components' of the machine may be people or computers; the 'routines' or 'procedures' could be carried out by humans, computers or both together.

Social machines are rapidly becoming a focus of computing research [1]. 'Programming the global computer' is one of the British Computing Society's grand challenges for computing, while peer-to-peer technologies have opened up the possibility of flexibly linking people and computers, as explored in projects such as OpenKnowledge (http://www.openk.org/) and the Social Computer community (http://www.socialcomputer.eu/).

Trust has always been recognised as an important factor in the function of such human/computer hybrids. However, the notions of trust used have often been relatively formal, imported from agent-based research. In this paper, I will examine the question of whether, and how, social computing can take into account wider and less well-ordered notions of psychologically realistic trust. I also note here two important limitations of scope of this paper. First, I focus here on issues of trust relevant to system designers fostering trust in their systems by users; of course there are many other stakeholders and many other trust relations typically involved (to take an obvious example, system designers have to trust users as well as being trusted by them). Secondly, I focus here on the challenges; solutions are already being created for these issues, but the point I want to emphasise in this paper is that we have to be clear about exactly how social machines rely on trust to function, and where a breakdown will lead to dysfunction. Without a precise model, it will be harder to diagnose problems.

[1] Electronics and Computer Science, University of Southampton, Highfield, Southampton SO17 1BJ, United Kingdom, kmo@ecs.soton.ac.uk.

2 SOCIAL MACHINES

In this section, I will flesh out the idea of a social machine or social computer. After a preliminary discussion, I shall briefly describe a couple of examples. A third subsection will examine the notion of programming social machines, before the section is completed with a brief sketch of the important role trust plays.

2.1 What is a social machine?

The idea of a social machine was implicit in early conceptions of the World Wide Web. As Berners-Lee put it in 1999:

Real life is and must be full of all kinds of social constraint – the very processes from which society arises. Computers can help if we use them to create abstract social machines *on the Web: processes in which people do the creative work and the machine does the administration.* ([2], pp.172, Berners-Lee's emphasis)

We see plenty of social machines around today. Many are embedded in social networks such as Facebook, in which human interactions from organising a birthday party to interacting with one's Member of Parliament are underpinned by the engineered environment. Another type of example is a multiplayer online game, where a persistent online environment facilitates interactions concerning virtual resources between real people. A third type is an online poker game, where the resources being played for are real-world, but where the players may be human or bots, and where the environment in which the game takes place is engineered around a relatively simple computational model. In such systems, (some of) the social constraints that Berners-Lee talks about, which are currently norm-driven, are converted to (or in his terms administered by) the architecture of the programmed environment.

These social machines are straightforward (*qua* interaction models), but as the technology is theorised more deeply it is inevitable that more complex systems will be developed. A generalised definition of a social computation is provided by Robertson and Giunchiglia:

A computation for which an executable specification exists but the successful implementation of this specification depends upon computer mediated social interaction between the human actors in its implementation. [3]

In such an environment, self-organisation (partial or full) becomes viable and scalable, while physical objects, agents, contracts, agreements, incentives and other objects can be referred to using Web resources (Uniform Resource Identifiers –

URIs). 'Programming' the social computer (rather than simply supporting and directing interactions on an engineered environment) and integrating larger numbers of people and machines will become increasingly feasible.

2.2 Examples

As a small example of a social machine, consider reCAPTCHA [4]. A CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart), invented by Louis Von Ahn, is the distorted sequence of letters that someone has to type in a box to identify him- or herself as a human (e.g. to buy a ticket online, or to comment on a blog). This is a task that computers cannot do, and so the system stops bots buying thousands of tickets for a concert or sporting event for later resale, or for a spambot to leave spam messages as comments to blogs.

Von Ahn extended the idea of the CAPTCHA to create the reCAPTCHA, which uses the same principle to solve another problem. Google (which acquired reCAPTCHA in 2009) wishes to scan and publish out-of-copyright books. However, Optical Character Recognition is too fallible to automate the process (in books over 100 years old, OCR fails for about 30% of words). The quantity of books to be scanned rules out human labour as a general solution to the problem.

Von Ahn noticed that his original CAPTCHA device was being used over 200m times a day, about half a million person-hours of effort. reCAPTCHA was designed to put these person-hours to more productive use. It presents the user who wishes to identify him- or herself as a human with two words, not one. The first is a normal CAPTCHA, and the second is a word from an old book that OCR had failed to identify. If the person succeeds with the first CAPTCHA, then he or she is known to be a human. As humans are reliable at word recognition, Google can therefore take the response to the second word as a plausible suggestion of what it is. Presenting the same word to multiple users allows a consensus to emerge.

The person is not necessarily aware that he or she is helping Google in its scanning task. The incentive for his or her involvement is the need for identification (to buy tickets, or comment on a blog, etc). The time taken for a reCAPTCHA is not significantly longer than a CAPTCHA. The 'machine' thereby created, of millions of people interacting via the reCAPTCHA facility, is currently identifying about 100m words per day (about 2m books equivalent per year). reCAPTCHA is offered as a free Web service to hundreds of thousands of websites (including Facebook, Twitter and Ticketmaster) which need spam protection; the service can be offered without a fee because of the translation service it also provides to Google [4].

As another example, Robertson and Giunchiglia [3] use the DARPA balloon challenge of 2009, in which all human 'components' of the machine are fully aware of their own role. In the DARPA challenge, the aim was to find ten weather balloons placed randomly around the US (in nine different states from California to Delaware). The rules of the challenge were intended to support the growth of a network of people taking part in the search, enabling a crowdsourced solution. The means of doing this in the winning solution (from Sandy Pentland at the Massachusetts Institute of Technology) was to set out financial incentives – everyone who discovered a balloon got a certain quantity of money, while for everyone who received a reward, the person who introduced them to the network received half that reward. Hence people were incentivised both to look for the balloons and to add more people to the network. Pentland's team began with 4 people, and using social media had recruited over 5,000 at the point of completion, which took under ten hours.

reCAPTCHA and the DARPA challenge were intended to solve a particular exogenous problem, but social machines can be designed to solve the problems of the people who constitute them. In such cases, the incentive of the participants is that the machine's smooth functioning is in their own interests. One could imagine, for instance, a set of computer-mediated interactions enabling a community to provide a social response to problems of crime (such as BlueServo, which crowdsources the policing of the Texas-Mexico border), or enabling those suffering from a particular health care problem to pool resources and to offer support and advice to fellow sufferers (such as curetogether.com). It will be obvious from these examples that such efforts will not always be uncontroversial.

Note finally that in many cases the ability to compute and to gather and process information at large scale is vital. This adds an extra layer of complication to the social machine vision.

2.3 Programming the social machine

Giunchiglia and Robertson define a social machine or computer as follows [3]:

A computer system that allows people to initiate social computations (via executable specifications) and adopt appropriate roles in social computations initiated by others, ensuring while doing so that social properties of viable computations are preserved. A general purpose social computer provides a domain-independent infrastructure for this purpose.

This implies three processes that need to take place in order for the social machine to run. First, specifications must be *initiated*, so that where necessary groups of people are able and willing to carry out parts of the computation. It may be that part of the 'programming' of the social machine will involve observation of and induction from existing social processes, to be adapted and reused in the new context of the social machine.

Second, people and groups must *adopt appropriate roles* in the machine, having been incentivised to join social computations. The *discovery* of these roles is an important issue.

Third, the groups relevant to the computation must be *reinforced*; as Robertson and Giunchiglia put it, "this relies on the computation being executed in a way that spreads the computation and knits together the social group via further social properties of the computation." In other words, the social computation must preserve the social structures necessary for its operation. In the example of the DARPA challenge, the clause that rewards anyone who has introduced a reward-winner gives incentives to people to add friends to an ever-growing network.

Robertson and Giunchiglia also define a *social property*, analogous to an *invariant* in conventional programming with real-world physical consequences: "a requirement associated with the specification of social computation that must be maintained, and perhaps communicated, during the execution of the specification in order for the computation to establish the social group needed to run it."

So if we return to the example of reCAPTCHA, its initiation involves publicising the Web service to sites needing spam protection, people adopt the appropriate role when they decide to solve a reCAPTCHA to get access to a service, and relevant

groups are reinforced by the success of the service in suppressing spam on sites to which people want access. The key social property to be preserved is that spam is suppressed; if spammers found an effective way around the reCAPTCHA, then fewer sites would support the Web service, and therefore fewer people would be playing the role of word recognisers.

2.4 The relevance of trust

Trust is essential to the smooth running of a social machine. Two precondition for social machines to motivate people to adopt appropriate roles is that they trust that promised incentives will appear, and that they trust that the machine will not do anything (in the world) that conflicts with their values. In the case of reCAPTCHA, people must trust that they will obtain access to their desired sites. In the case of the DARPA challenge, the participants must have trusted that the money would be paid out.

Trust is also central to the reinforcement of groups, as cooperation towards a goal demands trust in others' contributions; would Wikipedia authors bother to contribute if their work was routinely trashed without argued rationales? If an effect of a computation was to fragment the coalitions developed to carry it out by undermining trust between members, then it could not ultimately succeed. It is fair to say that for many social computations, trust (both between individuals in different roles, and of the machine by its component individuals) is likely to be a social property essential to the social machine's function.

Trust is of course most important when people take risks or place themselves in a vulnerable position with respect to a social machine. With reCAPTCHA this is barely an issue, but in a machine that, for example, enabled people to manage health care problems, users might need to pool information which could include sensitive health- or lifestyle-related data. That brings in complex rights-based issues such as privacy, and legal issues such as data protection.

In the next section, I shall briefly set out some of the most important properties of trust, as background to a discussion of issues that arise with respect to trust in social machines.

3. TRUST

The discussion of trust will be in four parts, beginning with an analysis of trustworthiness, upon which will be built an analysis of trust. Finally I shall discuss issues surrounding the connection of the two. These analyses are developed in more detail in a working paper [5].

3.1 Trustworthiness

Trustworthiness is prior to trust, which is an attitude toward the trustworthiness of others. Indeed, as Hardin has argued ([6], [7]), many commentators supposedly discussing trust are actually discussing trustworthiness. What, then, is this prior concept?

A trustworthy person is someone who does what she says she will do, all things being equal. This characterisation conceals quite a lot of structure. First of all, trustworthiness is a *property* of an *agent*. A *claim* must be made about her future actions. After all, it is absurd to accuse Barack Obama of being an untrustworthy brain surgeon, because he has never claimed to have brain surgery skills. The claim will also narrow the scope of trustworthiness; put another way, trustworthiness is context-dependent. The 'all things being equal' clause means that a trustworthy person need not succeed in carrying out the claimed behaviour, but if she does not, there must be an explanation for her failure which will absolve her of responsibility.

We can therefore define trustworthiness as a four-place relation, as follows:

(1) Y is trustworthy $=_{df}$ Tw<Y,Z,R,C>

where Y and Z are agents, R is a representation of the claim and C is a (task) context in which it applies.

In (1), Y is the agent who, if (1) is true, is trustworthy. R is the content of the claim made about her intentions, capacities and motivations for future behaviour. When (1) is true, Y's behaviour will be constrained by R. R may be explicitly written down, or may be implicit and understood; it may be open-ended and deliberately left unspecific to degrade gracefully. C is the set of contexts in which R is intended to apply (for instance, Y may claim to be a trustworthy car mechanic, but only within office hours, and only on certain makes of car).

This leaves Z, who is the agent responsible for generating and disseminating the claim R. In many, perhaps most, circumstances, Y = Z. However, this need not be the case. A trustworthy customer service employee (Y) respects a role description generated by her company (Z). A trustworthy piece of software (Y) performs according to a specification written by a designer (Z). It is essential that Z is *authorised* to make the claim about Y. Without authority, Z's claim has no bearing on Y's trustworthiness.

3.2 Trust

Trust is an *attitude* toward the trustworthiness of another. X trusts Y iff he believes that she is trustworthy (or, better, holds of the proposition 'Y is trustworthy' that it is true).

This characterisation of trust has a straightforward surface appearance. It is still a complex idea, however. Not only does trustworthiness import context-dependency, but trust forces us to confront a subjective element. There are six parameters of consequence in the trust relation, as follows:

(2) X trusts Y $=_{df}$ Tr<X,Y,Z,I(R,c),Deg,Warr>

with Y, Z and R as before, and X an agent.

In (2), the first three parameters are the relevant agents. X is the trustor and Y the trustee. Z, as before, is the agent who makes the claim R about Y's intentions, capacities and motivations. And again, as before, it could be that Z = Y (or, for that matter, X = Y, X = Z or X = Y = Z, although the possibility of these identities will not be defended here [5]).

Z makes a claim that Y's behaviour, all things being equal, will conform to R in contexts C. X's trust, if well-placed, should accept that claim. However, it need not, because X is only boundedly rational and communications between Z and X are not guaranteed to succeed. Furthermore, R might be implicit or unspecific. Hence X has to *interpret* R's meaning in the contexts in which he is interested. I have written this as a function I(R,c), to be read as X's interpretation of the force of R in the set of contexts that interest X, which I term c.

This brings trust's subjective aspect to the fore. For X's trust, it is *X's interpretation* that is the final arbiter, whether or not it is accurate. As trust is an attitude held by X about Y, it is X who supplies the underlying assumptions of the judgment. This has three specific consequences. First, for Y to maintain X's trust,

she must behave in accordance with I(R,c) *even if that differs from her own interpretation of R in c*. Second, for X to trust Y, it need not be the case that Z *has* authority to make claim R about Y. It is necessary only that X *believes that* Z has that authority. Third, I(R,c) only has any force with respect to Y if $c \subseteq C$, otherwise it will fall out of the scope of R. Yet for X's trust, it is necessary only that X *believes that* $c \subseteq C$. If any of X's beliefs is false – i.e. if the force of R in c is not I(R,c), or if Z does not have the authority to make claim R about Y, or if $c \nsubseteq C$ – X's trust or mistrust will be misplaced as based on a misunderstanding.

In short, in definition (2) above, X *believes that* (i) Z can authoritatively make claim R about Y, (ii) I(R,c) is the interpretation of R within a set of contexts c, and (iii) $c \subseteq C$.

This leaves two more parameters. Deg is a measure of X's confidence in his attitude toward Y's trustworthiness. The metric for Deg depends on the system under discussion. For psychological realism, it may be that Deg would be a fairly coarse-grained Likert-type psychometric scale of five or seven points. But it would be legitimate to produce more complex models that modelled Deg on, say, the real line between 0 and 1.

Whatever metric chosen must facilitate the expression of two types of trust judgment. First of all, X may have to choose whether he trusts Y_1 more than Y_2 to decide with whom to place his trust. Secondly, the level of risk that X takes on with respect to an interaction with Y will depend on his degree of trust; if he trusts her a lot, he will, all things being equal, be prepared to risk a lot, and if he trusts her only a little, his appetite for risk will be diminished.

Warr is the warrant for X's trust in Y. This could take any form – it doesn't have to be rational, and could even be that X has been dosed with oxytocin which increases the propensity to trust [8]. Unlike a warrant in Toulmin's system [9], the warrant explains the judgment, but is not intended for the persuasion of others. Nevertheless, usually there is a sensible rationale behind a trust judgment which is important for assessing it, and also for assessing how robust it is likely to be. Typical relatively reliable trust warrants include the reputation of Y, the past history of X's encounters with Y, the availability of sanctions for X, the possibility of a binding reciprocal agreement between X and Y, the credible commitments made by Y and the credentials that Y brings to the transaction.

As Wierzbicki argues ([10], pp.26-27), trust that does not have a rational component will be hard to model. That does not mean that trust cannot be irrational, but it makes it harder to embed psychologically-realistic trusting mechanisms into software, or to design sociotechnical systems (or social machines) which incorporate potentially irrational human trust judgments without restriction.

3.3 The problem of trust

The problem of trust is not to increase trust, but rather to ensure that X trusts Y when and only when Y is trustworthy. This is difficult as the incentives are not optimally aligned. If X risks assets in an interaction with Y, then he *benefits* from her *trustworthiness*, but unfortunately he only *controls* his *trust*. Conversely, Y benefits from X's trust, but only controls her trustworthiness. The result is a dilemma where the benefits of cooperation could be high, but losses to a trusting (trustworthy) party would accrue if their partner is untrustworthy (distrusting).

From this two things follow. First, trust cannot be an entirely rational attitude; as Hollis has argued, trustworthiness does not survive rigorous game-theoretic analysis (a fact available to rational would-be trustors) [11]. Second, X should use the analysis of (2) to determine where trust judgments can break down. Many failures of trust are down to differences in interpreting what Y is committed to.

A typical strategy for a trustworthy Y is to send *signals* of trustworthiness to X, which ideally will accurately represent her trustworthiness (would not be forthcoming if she were *not* trustworthy) and which will be included in X's warrant to trust Y [12]. These signals can be conscious or unconscious, and more or less strongly connected with the task that Y is offering to carry out, preferably as an unavoidable by-product. The flip side of any such signalling system, however, is that if it is made explicit, it can potentially be counterfeited by an untrustworthy person. Types of signal already mentioned include Y's reputation, history and credible commitments.

A second strategy involves structuring the encounter with some kind of *institution* (in the broad sense of a mechanism for producing order by structuring behaviour) which can reduce the likelihood of a deception being in Y's interest. Such an institution might supply objective credentials for Y, or might make plausible and effective sanctions available for X to apply if Y defects. Or X and Y might set up their own 'mini-institution' by entering into a reciprocal agreement. In each case, an institution promotes X's trust in Y only if X trusts the institution to deliver the structures it promises.

4 TRUST IN SOCIAL MACHINES: CURRENT APPROACHES

As noted earlier, trust is a vital element for social machines to function. However, this is a complex issue: in the open peer-to-peer architectures that will be required to support social machines, traditional knowledge engineering safeguards (such as centralisation of key functions, shared culture and ontologies, constraints and access control) are not practicable. In this section, I will expand on the theme of trust, using the theoretical apparatus assembled in Section 3.

Importing human interaction into the programming environment envisaged by Robertson and Giunchiglia presents a major challenge. Hendler and Berners-Lee see artificial intelligence as the key to enable people and machines to represent and reason over social attitudes including trust and trustworthiness, as well as related issues such as reliance and expectations; linked data and the Semantic Web will be important tools in such a world, by providing designers with access to a level of abstraction in which resources can be referred to directly and independently of the documents in which they are described [13]. Machines which require users to contribute information (such as those mentioned earlier to coordinate community responses to crime or healthcare issues) will also need to reason about privacy and data protection.

The human world is messy and full of compromise; computations in social machines must be able to cope with the consequences of this, such as inconsistency. Furthermore, given the sensitivity of personal data, social machines will also need to be able to function in hostile environments where some actors are malicious.

Although this is a lively area for research, there are few robust and scalable structures in place to represent these qualities. Hendler and Berners-Lee point out the importance of being able to treat these social phenomena as first-class objects capable of being reasoned over. The Semantic Web provides a blueprint for this, allowing the use of URIs to name objects of any kind [13].

In open environments, trust needs to be fostered from a number of sources. The most common view is to describe the relations between peers in a peer-to-peer architecture in terms of permissions and obligations governed by policies [14]. Theorem provers can determine whether peers have conformed to policies [15] and systems have been developed to explore the question of how to specify and verify strategies to determine whether and when to interact, and with whom [16].

5 DISCUSSION: THE HUMAN ELEMENT

One issue is that these approaches tend to assume that human trust behaviour is relatively well-behaved and if not rational at least fairly tidy and explicable. Yet as argued in section 3, it need not necessarily be so; as Kahneman has recently pointed out, rational processing coexists with fast, intuitive and emotional thinking [17]. Furthermore, the subjective element of trust is deep-seated. Hence policies may work very well to describe interactions in distributed systems unless elements are likely to behave idiosyncratically. Reasoning is only one approach to making a trust judgment, and may well involve a complexity that is inappropriate. Human judgments about trustworthiness of complex and distributed systems will not always align with the methods, ontologies and terms in which questions are framed by system designers. The key factors for consideration, as argued in section 3.2, include X's view of Z, X's interpretation of R, and the warrants that X accepts.

5.1 Displacing trust

Most approaches to trust in multi-agent systems assume that information relevant to agents' reputation, or data provenance, or data security will suffice to align trust and trustworthiness. Certainly transparency and availability of information about these is a bonus, and can do no harm. But will they be sufficient?

Trust is not always grounded; X's trust of Y may depend on his trust of Z. In many scenarios, X is given information by the system about the reputation of Y, or about the provenance of some information – it is widely accepted that these are important for trust. But even assuming that a typical X is willing to restrict his warrant for his trust in Y to reputation, provenance, recommendations and other mechanisms that have been extensively theorised online, he still needs to trust the *source* of the reputation/provenance/recommendation. If someone does not trust, say, Amazon, they are unlikely to trust the *-rating system that it hosts, even though it is intended to provide an objective assessment of Amazon's products. The provision of such information does not solve the trust problem – it just displaces it to another point of the system.

Recall also a point made earlier, that institutions can help promote well-placed trust if they are themselves trusted. It is also worth noting in this context that people contributing to a social machine, by trusting the machine's structuration of behaviour, also have to trust that their fellow users will behave in good faith. The trustworthiness of the machine will also depend on the trustworthiness of the user community. This is somewhat beyond the scope of this paper, which focuses on the challenges to designers, but the wide range of other stakeholders (owners, managers, shareholders, policymakers, users) should be an important focus of future research, and a complete social machine program should take all relevant roles into account.

5.2 The logic of trust

Z makes a claim about how Y will perform. Y in this case is the social machine, and Z the administrator. X's trust of the social machine will depend on his trust of the administrator. For instance, the motivation of the people from whom information is crowdsourced in the DARPA network challenge depended on financial incentives (a) to provide information to the administrator, and (b) to introduce new people to the group. The function of that social machine depended among many other things on enough people trusting the administration of the machine, and the likelihood of its dispersing the money.

Indeed, because we are dealing with trust with its subjective element, all that was required was that the various Xs *believed* that remuneration would be forthcoming. The money need not actually have been in place at all. Hence if we are formalising social machines using a process calculus (as advocated persuasively by Robertson and Giunchiglia), we need to make a distinction between those social properties which need to be *true* in order for a social machine to achieve its purpose, those properties which need to be *believed to be true* (but which need not be true), and those properties which need to be *both* true and believed.

This matters because a calculus should describe necessary conditions for a machine's function. In the case of the DARPA challenge, the existence of a pot of money to be distributed to the participants was *neither sufficient nor necessary* to the social machine's function. It was not sufficient, because if would-be participants were unaware of or did not believe in the financial remuneration they would not have taken part. It was not necessary, because all that mattered was that the participants were *motivated*, not that they were *paid*. Of course, this problem is most dramatic in a one-shot system, but will always re-emerge in some form even in contexts with repeat runs.

Indeed, spreading the truth about how a machine will function could on occasion undermine that very functioning. The reader may have noticed that someone helping Google by using a reCAPTCHA need not be aware that he or she is doing that (although Google makes no secret of it). This introduces an exploitative element to reCAPTCHA; one wishes to identify oneself as a human, but having done that, one is also required to perform an extra task, which is not identified as such, to help Google scan an old book.

reCAPTCHA demands very little effort, so the exploitation is probably bearable, but even so someone might resent having to help *Google* when they wanted to interact with *Facebook*. More generally, if people came to understand that, say, a social network was gathering information about them primarily in order to sell to marketing companies, or that a healthcare social machine was gleaning information primarily to sell to pharmaceutical companies, the feeling of exploitation (even if it was plausibly in the interests of the users) might have the effect

of discouraging the users from taking part. It is essential to make a distinction between what is known about the system, what users should believe (even if false) about the system, and what users should be unaware of (even if true) about the system.

5.3 Differences of interpretation

Where the interests of Z and X do not align, it is important to ensure that X's interpretation of R coincides with that of Z. This is not always the case with technology. Where Z is a designer who has created an artificial agent Y, Y's trustworthiness is often measured by Z against a highly technical specification R. However, the user X will typically see the technology holistically as part of a system with which he is confronted. If we take the example of an ID card, the system designer may be pleased to have devised a secure system. But the owner of the card will judge it in terms of the extent to which it empowers and constrains him. As Charles Raab puts it, "it is no comfort to a privacy-aware individual to be told that inaccurate, outdated, excessive and irrelevant data about her are encrypted and stored behind hacker-proof firewalls until put to use by (say) a credit-granting organization in making decisions about her" [18].

There are many types of case where R, the claim that is made about Y, can be very different from I(R,c), X's interpretation of that claim. If trust is to be maintained, R must be couched in a way that is meaningful for X. A merely technical specification of behaviour, however accurate, is unlikely to be enough. Yet a technical specification of the system's behaviour is required if we are to be able to program social machines rigorously.

6 CONCLUSION

The problem of trust is that it is hard to align to an arbitrary degree of certainty with trustworthiness. It is important, if dispiriting, to note that the most trustworthy system is useless if it is not trusted. Furthermore, it could happen that a trusted system works perfectly well (to its designers' satisfaction, anyway) *even if it is not trustworthy*.

Much will depend on the incentives given to participants. In the case of machines which provide a good user experience (for example, healthcare networking sites from which people get best practice or companionship or counselling from others with similar problems), specifying that experience will be difficult. All a designer can really specify are issues such as the privacy and security with which health data are stored. These are important factors for user trust, but the porousness of the system will also depend on the propensity of the networking humans to misuse or leak information they gain, for example from chatrooms. The nature of the user community is at least as important as the technical specification.

Taking this thought to a logical conclusion, it is likely that public trust in such machines will be highest when the public has had a say in their design and operation. The closer the relationship between trustor, designers and administrators, the better. This suggests that a focus of future research here might be the development of tools and protocols to allow communities to design social machines to their own specifications.

In machines such as reCAPTCHA and the DARPA challenge, where the humans in the loop are performing tasks subordinate to the wider goal of the system and gaining nothing intrinsic from participation, the classic trade-off of trust (that trust matters and trustworthiness is secondary, especially in one-shot games), is harder to avoid. 'Programming' of such machines using process calculi should, from the point of view of good design, make the necessary and sufficient conditions clear. Whether this promotes or restricts cynicism is an empirical question upon whose answer the future of social machines will probably rest.

ACKNOWLEDGMENTS

The work reported in this paper was funded by the EnAKTing project, EPSRC Grant EP/G008493/1. Thanks to Dave Robertson, Luc Moreau and three referees for comments.

REFERENCES

[1] A. Bernstein, M. Klein and T.W. Malone, Programming the Global Brain, *Communications of the ACM*, in press.

[2] T. Berners-Lee, *Weaving the Web: the Original Design and Ultimate Destiny of the World Wide Web*, Harper Collins, New York (1999).

[3] D. Robertson and F. Giunchiglia, Programming the Social Computer, *Philosophical Transactions of the Royal Society A*, in press.

[4] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham and M. Blum, reCAPTCHA: Human-Based Character Recognition via Web Security Measures, *Science*, 321:1465-1468 (12th Sept, 2008).

[5] K. O'Hara, A General Definition of Trust, working paper, http://eprints.ecs.soton.ac.uk/23193/, (2012).

[6] R. Hardin, Trustworthiness, *Ethics* 107:26-42, (1996).

[7] R. Hardin, *Trust*, Polity Press, Cambridge, (2006).

[8] M. Kosfeld, M. Heinrichs, P.J. Zak, U. Fischbacher and E. Fehr, Oxytocin Increases Trust in Humans, *Nature*, 435:673-676 (2nd June, 2005).

[9] S. Toulmin, *The Uses of Argument*, Cambridge University Press, Cambridge, 1958.

[10] A. Wierzbicki, *Trust and Fairness in Open, Distributed Systems*, Springer, Berlin, (2010).

[11] M. Hollis, *Trust Within Reason*, Cambridge University Press, Cambridge, (1998).

[12] A. Pentland, *Honest Signals: How They Shape Our World*, MIT Press, Cambridge MA, (2008).

[13] J. Hendler and T. Berners-Lee, From the Semantic Web to Social Machines: a Research Challenge for AI on the World Wide Web, *Artificial Intelligence* 174 156-161 (2010).

[14] M. Sloman, Policy Driven Management for Distributed Systems, *Journal of Network and Systems Management*, 2:333-360, (1994).

[15] M. Alberti, D. Daolio, P. Torrini, M. Gavanelli, E. Lamma and P. Mello, Specification and Verification of Agent Interaction Protocols in a Logic-Based System, *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC '04)*, ACM Press, New York (2004), 72-78.

[16] N. Osman and D. Robertson, Dynamic Verification of Trust in Distributed Open Systems, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India (2007), http://www.ijcai.org/papers07/Papers/IJCAI07-232.pdf.

[17] D. Kahneman, *Thinking, Fast and Slow*, Allen Lane, London, 2011.

[18] C.D. Raab, The Future of Privacy Protection, in R. Mansell and B.S. Collins (eds.), *Trust and Crime in Information Societies*, Edward Elgar Publishing, Cheltenham, (2005), 282-318.