How will software engineers of the Internet of Things reason about trust?

Andrew J. B. Fugard, Elke Beck, and Magdalena Gärtner

ICT&S Center, University of Salzburg
Sigmund-Haffner-Gasse 18, 5020 Salzburg, Austria
{Andy.Fugard, Elke.Beck, Magdalena.Gaertner2}@sbg.ac.at
WWW home page: http://icts.uni-salzburg.at

Abstract. The Internet of Things (IoT) will consist of everyday physical objects communicating with each other via massively distributed service-oriented architectures (SOAs). One neglected area of research is how engineers developing software underlying the IoT will decide whether the services they use and compose are trustworthy. We sketch how a formal socio-cognitive theory of trust can guide empirical research on the topic, and report preliminary results from 25 engineers who were asked how they currently reason about software component trustworthiness.

Keywords: trust, software engineers, components

1 Introduction

The Internet of Things (IoT) promises to be everywhere, from light bulbs and garden sprinklers to door locks and medicine bottles [4]. Artefacts on the IoT will use massively distributed and dynamically adapting service-oriented architectures (SOAs) [1] in which composable software services interact with each other via message passing. Given the pervasiveness of the things connected by the IoT, trust will be central. An important neglected question is: how will engineers of the software realising this future vision reason about the trustworthiness of the services they use and compose? For instance, how might a fire alarm software engineer trust software services for automatically unlocking doors in the event of a fire? The present paper addresses the question by empirically exploring how software engineers who use third-party components currently decide whether the components they choose are trustworthy. The main focus is on exploring how socio-cognitive processes beyond the technical SOA domain influence decisions made and impact on trust.

To investigate trust requires some theory of what trust is. According to Castelfranchi and Falcone [2], trust is a relation, $trust(X, Y, c, \tau, g_x)$, denoting that X (the trustor) trusts Y (the trustee) in context c to perform the tasks, τ, necessary to achieve X's goal, g_x. There are three main stages to trust in the theory:

1. The *evaluation* of the trustee on the basis of available information, including reputation and previous personal experience of the trustee. There are two

main types of evaluation, one involving implicit decisions (e.g., based on prior experience and routine), and the other a more explicit deliberation of reasons. The two types are not independent. Although it may be possible to find very general notions of trustworthiness, such as honesty and efficiency, the evaluation is always with respect to a particular goal.
2. An *intention* to delegate to the trustee. This is a positive evaluation, i.e., that Y is good for achieving the goals by carrying out the required actions.
3. The trust *decision*, a 'leap of faith' [6] whereby X delegates the tasks to Y and depends on Y., i.e., the actual action of trusting. This has happened when, for instance, the system goes live with a particular trusted service being used as an integral part.

Cognitive agents, defined as agents with goals and beliefs, play a special role in this theory: only they can be trustors, so, e.g., a software engineer can trust but a software service cannot; also whether a cognitive trustee is trusted can influence how trustworthy they are, whereas a software service cannot care. However the trustee (Y) need not be a cognitive agent, so, matching intuitions, it is possible according to the theory to trust a software component.

Most goals people want to achieve depend on tasks carried out by others. Correspondingly, people often work on tasks to enable others to achieve their goals. It might therefore come as no surprise that a major set of socio-cognitive processes concerns representing and drawing inferences about the goals of others [5]. We hypothesized that the trust decisions made by software engineers are influenced by a range of socio-cognitive factors beyond the technical domain. There is some anecdotal evidence of this, e.g., Dijkstra [3] argued (and warned) that software engineers often reason about software as if it had goals and intentions:

> The anthropomorphic metaphor is perhaps even more devastating within computing science [...]. Its use is almost all-pervading. To give you just an example: entering a lecture hall at a conference I caught just one sentence and quickly went out again. The sentence started with 'When this guy wants to talk to that guy...'. The speaker referred to two components of a computer network.

We designed a semi-structured questionnaire-based study to explore how software engineers choose software components and what impact their choices have on how trustworthy they think the component is.

2 Method

2.1 Participants

Participants were 25 (5 female) European software developers or designers. Their mean age was 34 years ($SD = 5.3$), with 4.7 years ($SD = 2.7$) in their current profession and 10 years ($SD = 5.1$) working in a software-related field.

Table 1. Types of goals engineers wanted to achieve

Goal type	*N*	*Example responses*
Functionality	19	'correct implementation of communication standards'
		'parse and store my data'
Reliability	6	'only do what I told it to do'
Performance	6	'time of response'
		'to be able to cope with a large number of queries'
Security	5	'not modifying data and not disclosing that data to third parties'
Consistency with documentation	4	'Work as expected based on the documentation'
		'be correctly implemented according to the . . . API'
Stability	1	'other alternatives were unstable'
Maturity	1	'other alternatives were . . . research prototypes'
Documentation clarity	1	'it had clear interfaces'
Simplicity	1	'A simpler solution would have been better'

2.2 Questionnaire

A web-based semi-structured questionnaire was developed. Participants were asked to consider a time they had chosen to use a software component. Questions concerning this choice were formulated to address each of the following entities and relationships: a software engineer (or set of engineers), X, trusts a software service (non-cognitive trustee), Y, developed by another engineer (or engineers; cognitive trustees) Z, to perform a sequence of tasks, τ, in context c, leading (X hopes), to the satisfaction of X's goal g_x. There are two trustees, one cognitive (the component developer) and the other not (the component). See the Appendix for the questions asked.

3 Results and Discussion

3.1 Software domains.

Participants chose examples from the following domains: semantic storage, banking, network management, data analysis, encryption, and simulation. These are representative of domains the IoT will involve. For instance a device might gather data from sensors, storing it in a semantic database or analyse it on the fly, and then send it encrypted across a network to a bank.

3.2 Engineers' goals.

Table 1 shows the main goals participants wanted to achieve. The most important concern was that the component achieves the participant's primary functional goals. This corresponds with results found elsewhere for non-engineers' uses of technology. Trustors have a primary goal, e.g., ordering a gift online, and issues such as reliability and security are secondary [7]:

Trustors initiate transactions with a primary goal in mind – order the birthday gift that is delivered in time, to pay a credit card bill before incurring a penalty. Discharging the tasks required to reach these goals in a secure manner – i.e. a way that does not put your credentials at risk – is a secondary goal.

3.3 Subjective trustworthiness.

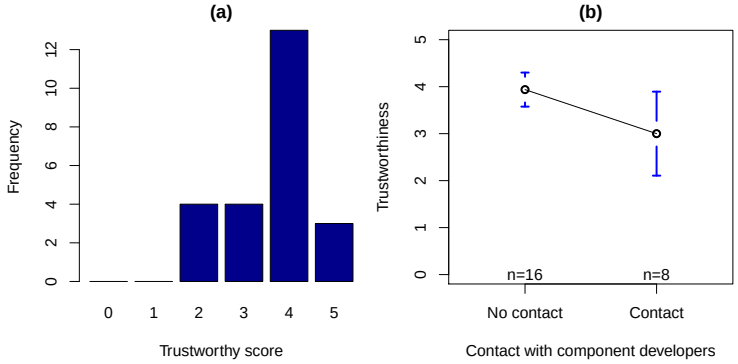

Fig. 1. (a) Distribution of trustworthiness ratings (0 = completely untrustworthy; 5 = completely trustworthy) and (b) mean trustworthiness (and 95% uncertainty intervals) as a function of whether the component's user had contact with its developers

Participants were asked, 'How trustworthy do you think the component is?' on a scale from 0 (= completely untrustworthy) to 5 (= completely trustworthy). A central claim of trust theories is that there must always be a 'leap of faith' [6], otherwise trust would be unnecessary. This was reflected in the responses people gave (see Figure 1(a)), with most respondents giving the assessment 4 out of 5. Several respondents chose to use a component, even though they gave it quite a low score (2 or 3). This illustates the distinction between trust *evaluation* and the trust *decision* introduced earlier. It is possible to delegate a task to a trustee even if inferred trustworthiness is low. In one case the component's results were always monitored, so the trustor's goals could still be achieved even though the trustee was seen as untrustworthy.

Given the important role of cognitive agents in the trust theory, we asked if participants had any contact with the component developers. We expected that contact would have had an impact on trustworthiness, for instance making it possible to increase perceived trustworthiness. Only 8 out of 25 participants answered in the affirmative. Contact was via mailing lists and forums (3), email (3), bug tracking systems (1), and training courses (1). Contrary to expectations, those who did have contact gave a slightly lower trustworthiness evaluation than those who did not ($t(9.9) = 2.3$, $p = .047$; non-parametric Wilcoxon rank sum

test $W = 94$, $p = .048$; see Figure 1(b)). There is a plausible explanation for this difference: all of the contact concerned problems, e.g., problems installing software, incompatibility between different library versions, bug reports, so, in this sample, the act of initiating contact was a signal of untrustworthiness.

3.4 Processes to infer trustworthiness.

There were two main technical approaches mentioned. Unsurprisingly, various types of software testing were most important. Testing is a technical analogue of experience building seen when non-engineers decide to trust [6]. Occasionally source code was also inspected.

Two non-technical approaches were mentioned. Chatting with colleagues was seen as important, e.g., other team members, project partners, and the boss or project/team leader. This illustrates another case of delegation, that of the very decision to trust. Engineers trust their peers and team leads to help them choose which components to trust. Finally reputation was important, e.g., positive feedback on developer forums, a component being 'well-established'.

4 Conclusions

Guided by a formal trust theory, we explored how software engineers think about trust. As hypothesized, in addition to using technical approaches, software engineers rely on the same basic socio-cognitive processes as do non-engineers, e.g., reasoning informally about reputation, making a 'leap of faith', focusing on primary functional goals. Better understanding the role of these socio-cognitive processes will help ensure the Internet of Things is trustworthy. Future work should investigate how interactions between service producers and consumers might have a positive influence on trust, e.g., by interaction design enabling positive contact when everything is working as hoped in addition to queries in the event of problems with a service.

Acknowledgements. This work was funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant 257930 (ANIKETOS; see `http://www.aniketos.eu/`).

References

1. Atzori, L., Iera, A., Morabito, G.: The Internet of Things: A survey. Computer Networks 54(15), 2787–2805 (2010)
2. Castelfranchi, C., Falcone, R.: Trust Theory: A Socio-cognitive and Computational Model. John Wiley and Sons Ltd, Chichester, UK (2010)
3. Dijkstra, E.W.: On anthropomorphism in science. Tech. rep. (1985), http://www.cs.utexas.edu/users/EWD/ewd09xx/EWD936.PDF
4. Gershenfeld, N., Krikorian, R., Cohen, D.: The Internet of Things. Scientific American 291(4), 76–81 (2004)

5. Kilner, J.M., Friston, K.J., Frith, C.D.: Predictive coding: an account of the mirror neuron system. Cognitive Processing 8, 159–166 (2007)
6. Möllering, G.: Trust: Reason, Routine, Reflexivity. Elsevier, Oxford (2006)
7. Riegelsberger, J., Sasse, M.A.: Ignore these at your peril: Ten principles for trust design. In: Proceedings of Trust 2010: 3rd International Conference on Trust and Trustworthy Computing (2010)

Appendix: Semi-structured questionnaire

1. Consider a time you have chosen to use a software component. What was its function?
2. In what programming language(s) was it written (leave blank if you don't know)?
3. 'Trust' and 'trustworthiness' have many different meanings. In the context of the software component you have in mind, what exactly did you trust it to do (or not to do)?
4. Did you have any contact with the developers of the software component? [Yes/no.]
5. How did you get in contact?
6. Why did you communicate with the developers? Briefly describe what you discussed.
7. What kind of software did the component become part of?
8. Approximately how many people (including you) worked on the project?
9. Why did you choose to use the selected component and not another component? Briefly describe any reasoning behind the decision.
10. With whom did you speak to help decide whether to use the component? (Leave blank if nobody.)
11. Did you perform any kind of real-time monitoring or checking of the component in the live project?
12. Please describe the real-time monitoring or checking that was performed.
13. Is there a situation or context in which you would not trust the component? [Yes/no.]
14. In what situation/context?
15. How trustworthy do you think the component is? [Rated from 0 = completely untrustworthy to 5 = completely trustworthy on a discrete scale.]
16. What (if anything) would have changed your mind about the trustworthiness of the component?
17. What would the consequences have been if the component had turned out not to be trustworthy? (Try to be as specific as possible.)
18. In retrospect, is there anything you believe you should have done to decide how trustworthy the component was?
19. Do you have any general comments you think might be of relevance?