Academic Panel: Can Self-Managed Systems be trusted?

Rogério de Lemos[*], Julie A. McCann[%] (editor), Omar F Rana[@] and Andreas Wombacher[#]

[*] *Computing Laboratory, University of Kent, UK, r.delemos@kent.ac.uk*
[%] *Department of Computing, Imperial College London, UK, jamm@doc.ic.ac.uk*
[@] *School of Computer Science, Cardiff University, UK, O.F.Rana@cs.cardiff.ac.uk*
[#] *Department of Computer Science, University of Twente, NL, A.Wombacher@utwente.nl*

1. Introduction.

Trust can be defined as to have confidence or faith in; a form of reliance or certainty based on past experience; to allow without fear; believe; hope: expect and wish; and extend credit to. The issue of trust in computing has always been a hot topic, especially notable with the proliferation of services over the Internet, which has brought the issue of trust and security right into the ordinary home. Autonomic computing brings its own complexity to this. With systems that self-manage, the internal decision making process is less transparent and the 'intelligence' possibly evolving and becoming less tractable. Such systems may be used from anything from environment monitoring to looking after Granny in the home and thus the issue of trust is imperative.

To this end, we have organised this panel to examine some of the key aspects of trust. The first section discusses the issues of self-management when applied across organizational boundaries. The second section explores predictability in self-managed systems. The third part examines how trust is manifest in electronic service communities. The final discussion demonstrates how trust can be integrated into an autonomic system as the core intelligence with which to base adaptivity choices upon.

2. Self-management Applied across Organizational boundaries - Andreas Wombacher.

Integration of information systems is performed on different organizational levels: the intra-organizational integration means to couple systems which are owned and maintained within a single organizations by different departments or business units, while cross-organizational integration results in B2B systems allowing business to coordinate their processes using information systems. While the requirements on trust and security are much higher at cross-organizational integration, the underlying processes of implementing the integration are similar. In particular, right now, the implementation process works like this: people meet, discuss different options of the integration, decide on a particular one, drill it down to the different parts to be provided by the different parties, and afterwards implement the agreed specifications. This centralized process is expensive and time consuming, but provides sufficient options to pay attention to personal and trust relationships. However, this kind of integration is appropriate for long running business relationships, which result in stable information system interfaces provided by the different parties as a basis for the integration.

Opposed to this static approach of integration, Service Oriented Architectures (SOA) provide a set of standards for communication and the loosely coupling of services. Services, in particular, are offered and maintained by a service provider, thus are maintained in a decentralized way. The loose-coupling supports integration of systems with lower costs supporting also short-term integration of information systems. As a consequence, the process of integration can be addressed in a more bottom-up way where services are provided first and coupling of services is performed in case it is needed. The loose-coupling allows ad-hoc composition of services implementing system integration in a cheap and fast way.

Inspecting the way SOA is applied in current applications indicates that they make use of the common communication methods, but do not use the potential of loosely-coupling, that is, the bottom-up integration. In particular, the decentralized maintenance of services has not been established right now. That is, decisions on changing the functionality, the Quality of Services (QoS) parameters, or the provided options of a service (process model) are made by the service provider, but are not necessarily be based on the observed requirements of his service users. In particular, the service development process has to be changed from a centrally coordinated one to a decentralized, market driven one. Market-driven means that the success of a service offering is measured, e.g., monetary, in the number of parties using the service, or the number of invocations of a service in average. If software is developed in this decentralized and market driven way, the change of properties and functionality of

services is more dynamic and communication of these changes gets more complicated.

Such a change in the service development process has also implications on the systems derived from the system integration process. A potential change of a service may not affect the integrated system at all, may require a slight change of other services provided by third parties, who are willing to apply these changes, or may result in a termination/cancellation of the integrated system in case other service providers are not willing to adapt the required changes. Since services are changing more often, an integrated system requires also adaptation of the way it is integrated and maintained to remain operative. This continuous adaptation is clearly specified, is quite easy to automate and has a clear optimisation function, which makes it a good candidate for self-management.

Self-management in system integration based on SOA means that a service provider doesn't have to do the integration processes himself, that is, initiating the lose-coupling of the services, but use service provisioning system providing this functionality to him. In particular, the self-managed service provisioning must support the initialisation of an integrated system as well as its maintenance especially keeping track and adapting to changes initiated by all kind of changes either locally or by external service providers.

Applying this concept to e.g. Web Services as a potential infrastructure of SOA, it turns out that the technology provided so far is not mature enough. In particular, the following issues can be observed:

- the service discovery of state dependent services does not cover process, QoS, and semantic aspects in an appropriate way, although a lot of work is currently going on in this area
- the decentralized decision making on consistency of an integrated system with regard to cover process, QoS, and semantic aspects has only been addressed partially
- self-managed fault-tolerance and recovery of SOA based integrated systems is not supported right now, although approaches exists which are mainly applicable to services with a centralized coordinator
- decentralized, efficient and reliable auditing of the execution of a SOA based integrated system has not really been addressed in research community
- change management of processes, QoS, or semantic aspects of a service and its application to the currently running SOA based integrated system instances is another open issue

Due to these exemplary limitations derived from the underlying infrastructure, the risk of causing a non-working integrated system is quite high, which results in high costs due to the nun-operational integrated system. Thus, the willingness of system integrators to consider self-managed SOA based integrated systems is quite limited. However, a big community of researchers are working on the specific areas mentioned above to cover up the technological issues, thus it is worth digging into applications of self-management approaches to SOA based integrated inter- and cross-organizational systems.

3. Predictability in Self-Managed Systems - Rogério de Lemos.

Trust can be defined as the reliance put by a system on some properties of another system [8]. Consequently, a trusted system has a set of properties that are relied upon by another system, i.e., there is an accepted dependence. This concept of accepted dependence is dependence allied to the judgment that the level of dependence is acceptable [1], and this level of dependence can vary from total dependence to complete independence. For total dependence, for example, the failure of one system might cause the failure of another system that relies upon its services. From a narrow perspective, dependence might be associated to the services correctness delivered by a system [3,7]. However, dependence, like trust, cannot be considered in absolute terms, instead it should be expressed by a set of properties.

A *self-managing system* can be seen as a system that has the ability to react and adapt dynamically to either internal or external changes without any outside interference. Whether a self-managed system can be trusted relies on the level of dependence that another system places upon it, and not necessarily on its services or quality of its services. If the effect of self-management activities are completely transparent to the services that the system delivers, then whether the system is self-managed or not should not affect the level of trust that other systems place upon it. On the other hand, if the effect of self-management activities is reflected on the services that the system delivers, then uncertainties might appear on its services depending on the techniques employed for implementing the self-managing capabilities [4].

Techniques for implementing the self-managing capabilities essentially depend on how a system is described: process or data [6,9]. *Process description* characterises the system as sensed by providing the means for producing or generating objects having the desired characteristics [9]. *Data description* characterises the system as acted upon by providing the criteria for identifying objects, often by modelling the objects themselves [9]. The difference between process and data representations can be interpreted from the perspective of accuracy and precision. While uncertainties in process descriptions can be eliminated within a certain degree of confidence, in data descriptions, these are difficult to be eliminated.

If trust is based on the predictability of the services required from a self-managed system, then in the context of process descriptions this could be achieved, as it is achieved in most of the existing dependable systems. In contrast, in the context of data descriptions, the elimination of uncertainties for obtaining predictable services might not be as desirable, since it would remove a feature that could be essential for the provision of adaptability. Clearly, a trade off is established between the self-management capabilities of a system and the trust that other systems place upon that system. For example, let us assume a system composed of several self-managed components whose behaviours are not entirely predictable. If no constraints are imposed on the interactions between these components, it would be difficult to establish overall system emergent behaviour, considering the uncertainties associated with the behaviours of the individual component. Moreover, the emergent behaviour might be either beneficial or harmful, how to incorporate means that are able to clearly identify and promote which is which, still remains a challenge.

From the practical point of view of a system based on *process* description, let us consider dynamic reconfiguration as a mechanism for handling replication in self-managed systems [5]. Assuming that only one fault might occur, the defined architectural solution compares two streams of data for implementing crash failure semantics (processing halts before an incorrect outcome is produced). With a certain degree of confidence, which depends on the veracity of the fault assumption, trust can be obtained that the service delivered by the system will be correct if enough replicated resources are available. For a system based on *data* description, let us consider anomaly detection as the first stage for self-management in the context of a hierarchy of dependable embedded systems [2]. Systems at the lower level have the capability for identifying new anomalies, and incorporate corresponding detectors into their repertoire of detectors. Whether these new detectors should be distributed among other systems, as an analogy to vaccination, it is not yet an automatic process since it requires them to be validated by a domain expert.

In conclusion, considering predictability as a major criterion for placing trust upon a self-managed system, data description solutions seem promising for emerging complex applications that are open and collaborative in their nature, however they still lack in providing the necessary assurances in the context of dependable applications.

4. Trust-based Electronic Service Communities - Omer F. Rana.

The general notion of “trust” is excessively complex, and appears to have many different meanings depending on how it is used. There is also no consensus in the computer and information sciences literature on a general definition of “trust” – although its importance has been widely recognized in the increasing number of publications that utilize it. Trust is also a multifaceted issue and may be related to other themes such as risk, competence, security, beliefs and perceptions, utility and benefit, and expertise. Hence, a service user may only be interested in evaluating the trust of a service provider if there is likely to be some risk to the service user directly. Overall, two main approaches may be deduced from literature. The first is based on allowing “agents” in a system to trust each other and therefore there is a need to endow them with the ability to reason about the reliability or honesty of their counterparts. This ability is captured through trust models. The second is based on allowing agents to calculate the amount of trust they can place in their interaction partners. This is achieved by guiding agents in deciding on how, when and who to interact with. An agent in this context refers to either a service user or a provider. However, in order to do so, trust models initially require agents to gather some knowledge about their counterparts. This may be achieved in three ways:

- A Presumption drawn from the agent's own experience: trust is computed as a rating of the level of performance of the trustee. The trustee's performance is assessed over multiple interactions to check how good and consistent it is at doing what it says it will [1,12]. This aspect of trust may utilize a pre-agreed contract between a service user and provider. Violation of a contract is likely to impinge on the trust that the service user has in the provider.
- Information gathered from other agents: trust in this approach is computed indirectly from recommendations provided by others. As the recommendations could be unreliable, the agent must be able to reason about the recommendations gathered from other agents. The latter is achieved in different ways: (1) deploying rules to enable the agents to decide which other agents' recommendation they trust more; (2) weighting the recommendation by the trust the agent has in the recommender [13,14]. Such a referral mechanism may involve a multi-hop interaction.
- Socio-Cognitive Trust: trust in this case is the capability to characterize the likely motivations of other agents. This involves forming coherent beliefs about different characteristics of these agents, and reasoning about these beliefs in order to decide how much trust should be put in them [5].

Client agents involved in an autonomic system may need to choose between a set of partners to interact with. This situation arises when a client requires a service, and multiple providers are available to offer such a service. Selection between such a set of providers incurs a

computational cost – which may increase as the number of interactions and service execution requests increase. We therefore define the concept of a "community", within which participants can interact with a higher level of trust on each other. In this instance, trust may be viewed with reference to each of the three criteria mentioned above. A reason for the formation of such communities may be to reduce the subsequent cost of interaction once the community has been established. An agent may therefore decide to incur an initial cost to determine which community it should participate in, what actions it should undertake within the community (its role), which other participants it should communicate with (its interactions), and when to finally depart from the community. Based on such an analysis, an agent would have to pay an initial cost to make some of these decisions. Subsequently, the agent will only incur an "operational" cost -- much lower than that for making some of these initial decisions. Formation of such communities may be implicit – i.e. based on analysing interactions of agents and determining the level of trust that can be placed on other participants, or explicit – i.e. a system administrator may determine which set of agents must be placed in a community. We demonstrate implicit service communities, and use this as a basis to evaluate trust. Communities, by definition are dynamic in nature, and will have a varying membership.

5. Integrating Trust as Autonomic Intelligence – Julie A McCann[1].

The paradigm shift in computing required to achieve Mark Weiser's vision of *Calm* for ubiquitous computing lies in the emergent intelligence embedded in the autonomic middleware governing it. Context-awareness is the ability of an application to adapt itself to the context of its user(s), whereby a user's context can be defined broadly as the circumstances or situations in which a computing task takes place. One of the most common contexts is the location of the user (or of objects of interest). In smart-homes, location can be obtained using a variety of different alternative sensor types, including ultrasonic badges, RFID-tags, and even pressure sensors in the floor. The quality (which is quantitatively application-specific) of the location information acquired by different sensors will however be different. For instance, ultrasonic badges can determine location with a precision of up to 3 cm, while RF lateration is limited to 1–3 m. Thus, we can define properties, which we call Quality of Context (QoC) attributes that characterise the quality of the context data received. QoC is essential to choosing the best alternative among those available when delivering a specific type of context. Therefore this discussion focuses on the intra- dependences within the autonomic systems services. Context providers need to specify QoC attributes for the context information they deliver. These attributes may vary over time and therefore must be updated regularly. But how can we trust the QoS advertised by each service? This section briefly introduces a method that allows an autonomic system to discern between service contexts while tracking the published quality rating [10].

An application makes use of a number of context services that in turn use one or more context providers. The application defines the minimum QoC required for correct functioning. We define an application's satisfaction for a particular CP as a utility function that maps the CP's QoC attributes to a value that quantifies the application's satisfaction (where values >= 0 mean the application is satisfied with this CP and values < 0 mean the application is not satisfied). Given each application's idea of the CP utility it requires in the form of its utility function, the adaptation engine can then apply each application's utility function to each CP and select for each application the best CP. For example, the utility function could define a reference point of the application's QoC expectations and then use linear distance (1-norm), Euclidean distance (2-norm), or return the maximum distance (maxnorm) between a CPs provided QoC and an application's QoC expectation. This way we can determine an application's satisfaction (or dissatisfaction) when the CP is unable to provide an estimate of a QoC attribute, given the value wished for by the application. The various norms won't usually be applied but one also considers the sign (satisfaction (+) vs. dissatisfaction (-)) of each dimension and also of the final distance. Since each QoC attribute is not necessarily equally important, e.g. precision may be more important than refresh rate, applications may set weights to the QoC attributes.

We believe that among the descriptive attributes of a CP that are used as input to an application's utility function, there should also be a measure of the CP's trust. In our middleware, we define the trust of a CP as *the probability that, when it delivers context information, the quality of this information will match the descriptive attributes advertised*. Therefore, if a location precision of 10cm is advertised, but the actual location is 50cm from what is delivered by the CP, then the CP is being unreliable. Instead, a CP advertising a location precision of 100cm but delivering information within 50cm of the actual location is dependable. Including trust in the input of the utility function allows an application to choose how much risk it is willing to take in the hope of receiving good quality information. Trust (from now on abbreviated as *tw*) is different from all other descriptive attributes in that it cannot be determined by the CP itself (which could

[1] Based on work carried out by Markus Huebscher with Julie A McCann (see [10]).

otherwise choose maximum *tw*=1), but must be determined externally. We use a learning model that takes as input binary positive/negative feedback from context consumers and cross-validation with other CPs and feeds this feedback into a parameterised probability density function that is used to predict the CP's current trust. The model allows for dynamic trust by keeping a window of recent feedbacks that affect the learning model. Thus, should the ratio of positive/negative feedbacks change over time, and then so will the predicted *tw* of the CP.

Concluding trustworthiness as one of the metrics in the utility function deciding which alternative provider to pick, we have found in an experimental case study that the resulting output from the middleware is not only as good as the best alternative, but even better, as the middleware switches continuously to the current best. This is a result of the fact that no alternative is the trustworthiest all the time. As trustworthiness is not necessarily a result of malicious intent, it is volatile and dynamic.

6. Conclusion.

Through the exploration of trust from as diverse approaches as the examination of trust cross-organizationally and in electronic service communities, its predictability and finally, trust as the core of the self-managing system, we can see that the issue is quite large and complex. We have shown that in each domain we continue to observe technological advances aiming to help solve the problems of trust in autonomic systems and have identified that trust is possibly key to the take-up of self-managing systems in the future. However this is closely tied to the provision of assurances as we see perhaps that assurance can only be safely given within closed technological and/or organisational communities (initially at least).

7. References.

1. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. "Basic Concepts and Taxonomy of Dependable and Secure Computing". *IEEE Transactions on Dependable and Secure Computing 1(1)*. January-March 2004. pp. 11-33.
2. M. Ayara, J. Timmis, R. de Lemos, and S. Forrest. "Immunising Automated Teller Machines". Submitted to the 4th International Conference in Artificial Immune Systems (ICARIS 2005). Banff, Canada. August 2005.
3. F. Cristian. "Understanding Fault-Tolerant Distributed Systems". *Communications of the ACM 34(2)*. 1991. pp. 56-78.
4. R. de Lemos. "The Conflict between Self-* Capabilities and Predictability". *Self-Star Properties in Complex Information Systems*. O. Babaoglu, M. Jelasity, A. Montresor, C. Fetzer, S. Leonardi, A. van Moorsel, M. van Steen (Eds.). Lecture Notes in Computer Science 3460. Springer. Berlin, Germany. 2005. pp. 219-229.
5. R. de Lemos. "Architecting Web Services Applications for Improving Availability". *Architecting Dependable Systems III*. R. de Lemos, C. Gacek, A. Romanovsky (Eds.). Lecture Notes in Computer Science 3549. Springer. Berlin, Germany. 2005 (to appear).
6. G. J. MacFarlane. "Information, Knowledge and Control". *Essays on Control: Perspective in the Theory and its Applications*. H. L. Trentelman, J. C. Willians. Birkhäuser (Eds.). 1993.
7. D. Parnas. "On the Criteria to be used in Decomposing Systems into Modules". *Communications of the ACM 15(12)*. December 1972. pp. 1053-1058.
8. D. Powell, and R. Stroud (Eds.). *Conceptual Model and Architecture of MAFTIA*. MAFTIA deliverable D21. IST Project on Malicious- and Accidental-Fault Tolerance for Internet Applications (MAFTIA). January 2003.
9. H. A. Simon. *The Sciences of the Artificial*. Second Edition. MIT Press. Cambridge, MA, USA. 1981.
10. M C. Huebscher, and J A. McCann. "A Learning Model for Trustworthiness of Context-awareness Services". *Proceedings of the 3rd International Conference on Pervasive Computing and Communications (PerCom)*. March 2005.
11. J. Sabater, and C.Sierra, "REGRET: A Reputation Model for Gregarious Societies". *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agents Systems*. 2002.
12. M. Witkowski, A. Aritikis, and J. Pitt. "Experiments in building Experiential Trust in a Society of objective-trust based agents". *Trust in Cyber-societies*. 2001.pp. 111-132,
13. S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The Eigentrust Algorithm for Reputation Management in P2P Networks". *Proceedings of the Twelfth International World Wide Web Conference*.2003.
14. L. Page, S. Brin, R. Motwani, and T. Winograd. *The PageRank Citation Ranking: Bringing Order to the Web*. Tech. Report SIDL-WP-1999-0120, Stanford Digital Library Technologies Project, Stanford Univ., 1999; http://dbpubs.stanford.edu:8090/pub/1999-66.
15. R. F.C.Castelfranchi, "Principles of Trust Formas: Cognitive Anatomy, Social Importance and Quantification" *Proceedings of the International Conference on Multi-Agent Systems* 1998.