The Knowledge Engineering Review, Vol. 21:3, 239–248. © 2006, Cambridge University Press
doi:10.1017/S0269888906000853 Printed in the United Kingdom

Can self-managed systems be trusted?

Some views and trends

JULIE A. MCCANN,[1] ROGERIO DE LEMOS,[2] MARKUS HUEBSCHER,[1] OMER F. RANA[3] and ANDREAS WOMBACHER[4]

[1]*Department of Computing, Imperial College London, UK;*
e-mail: jamm@doc.ic.ac.uk
[2]*Computing Laboratory, University of Kent, UK;*
e-mail: r.delemos@kent.ac.uk
[3]*School of Computer Science, Cardiff University, UK;*
e-mail: o.f.rana@cs.cardiff.ac.uk
[4]*Department of Computer Science, University of Twente, NL;*
e-mail: a.wombacher@utwente.nl

Abstract

Self-managing or autonomic computing is a highly emerging field having been championed by industry and academia alike. Essentially the management of the system is being handed to the system itself, therefore the issue of trust in terms of the decisions the system makes regarding itself, is of paramount interest. This paper considers four aspects of trust in self-adaptive computing systems with the aim to identify trends and pinpoint areas that require more investigation.

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. In this paper we use the general definition of trust rather than specifically trust meaning security, as some computing fields would define it. The issue of trust has always been important to computer scientists, especially notable with the proliferation of services over the Internet, which has brought the issue of trust and computer security right into the home. Autonomic computing brings further 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. Therefore we begin to question the extent to which we trust the systems managing our systems, as we become more hands-off. Self-management is being used in systems such as Web services, run-time database optimization for anything from environment monitoring to looking after granny in the home and thus the issue of trust is imperative.

To this end this paper brings together authors who work on diverse aspects of autonomic computing 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.

https://doi.org/10.1017/S0269888906000853 Published online by Cambridge University Press

2 Self-management applied across organizational boundaries

Integration of information systems is performed on different organizational levels: intra-organizational integration means to couple systems which are owned and maintained within a single organization by different departments or business units, while cross-organizational integration results in B2B systems allowing businesses 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, and 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 loose-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 also supporting 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 common communication methods, but do not use the potential of loose-coupling that relies on 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 options provided by a service (process model) are made by the service provider, but are not necessarily 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, in monetary terms, in the number of parties using the service, or the number of invocations of a service on 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 also has 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 also requires 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 optimization function, which makes it a good candidate for self-management.

Self-management in system integration based on SOA means that a service provider does not have to do the integration processes himself, that is, initiating the loose-coupling of the services, but uses a service provisioning system providing this functionality to him. In particular, the self-managed service provisioning must support the initialization of an integrated system as well as its maintenance especially keeping track and adapting to changes initiated by all kinds 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:

https://doi.org/10.1017/S0269888906000853 Published online by Cambridge University Press

- 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 exist 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 the 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 non-operational integrated system. Thus, the willingness of system integrators to consider self-managed SOA-based integrated systems is quite limited. However, a large 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

In this section, we argue that if the notion of trust depends on how predictable the behaviour of a self-managing system is then such trust can only be obtained from systems that are based on process descriptions. However, for systems whose behaviours are intrinsically uncertain, such as those based on data descriptions, such trust can nevertheless be achieved if safetymargins are imposed upon the behaviour of the system for the purpose of avoiding undesirable behaviours. But before we delve into the differences between systems based on process and data description, trust and self-managing systems are defined.

Trust can be defined as the reliance put by a system on some properties of another system. 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 allied to the judgment that the level of dependence is acceptable (Avizienis *et al.,* 2004), 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 related to the correctness of services delivered by a system (Parnas, 1972; Cristian, 1991). However, dependence, like trust, cannot be considered in absolute terms, instead it should be expressed by a set of properties. Examples of such properties are the dependability attributes like reliability, availability, confidentiality and integrity, where dependability is related to the ability of a system to deliver service that can justifiably be trusted. Such justification can be the provision of assurances that the behaviour of a system is predictable, i.e. the behaviour of a system can always be inferred under known operational conditions.

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. The process of reacting and adapting can be reflected in the actual services delivered by the system, or remain completely transparent from the delivered services. An example of the former would be a system that changes its behaviour in order to react to changes that occur in its environment, such as, a high level of fraud might enforce an upper limit on the amount of cash an ATM is able to dispense. An example of the latter would be a system that needs reconfiguring of its architecture in order to accommodate a failure in one of its components, such as, a failure in the printing mechanism of an ATM should not hinder the dispensing of money.Whether a selfmanaged system can be trusted

https://doi.org/10.1017/S0269888906000853 Published online by Cambridge University Press

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 self-management capability of a system does not affect the services it delivers and their quality, in a way that reliance placed upon it is not compromised, then the self-managed capability should be transparent towards the level of trust. On the other hand, if the self-management capability is reflected in the delivered services and their quality, in a way that reliance can be compromised, then trust should be affected. Whether the self-management capabilities affect reliance depends on the techniques employed for realising the self-managing capabilities. Some of these techniques, due to their nature, introduce uncertainties on the services delivered by the system, and these uncertainties might influence the trust that other systems place on a self-managed system (de Lemos, 2005). For example, these techniques could be based on unknown emergent behaviours that might be undesirable for other systems that depend on them.

Techniques for implementing the self-managing capabilities essentially depend on how a system is described: process or data (Simon, 1981; MacFarlane, 1993). *Process description* characterizes the system as sensed by providing the means for producing or generating objects having the desired characteristics (Simon, 1981). *Data description* characterizes the system as acted upon by providing the criteria for identifying objects, often by modelling the objects themselves (Simon, 1981). The difference between process and data representations can be interpreted from the perspective of accuracy and precision.While process descriptions might be precise but not accurate, data descriptions might be accurate but not precise. In process descriptions, the assumptions that allow a process to be realizable introduce uncertainties. However, whenever these assumptions are discharged, more accurate models can be obtained, thus eliminating uncertainties fromthe system. Trust can then be justified based on arguments that assumptions are unlikely to be violated, for example. On the other hand, since data descriptions are abstractions of the actual system behaviour, these descriptions cannot be precise because of the uncertainties associated with the data. Moreover, in some cases, for the data to be meaningful, it has to be processed, which introduces further uncertainties to the system description. In summary, while uncertainties in process descriptions may be eliminated within a certain degree of confidence, in data descriptions, these are difficult to eliminate because of their intrinsic nature. However, the latter can be useful for providing the basis for systems to adapt to changes, mainly those that are unexpected, through emergent behaviours (de Lemos, 2005).

If trust is based on the predictability of the services required from a self-managed system, i.e. the lack of uncertainties, then in the context of process descriptions this could be achieved, for example, through the mathematical description of a system. 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 system features that could be essential for the provision of adaptability. Clearly, a trade-off is established between the self-management capability and the behavioural predictability of a system. For example, let us assume a system is 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 what would be the overall system-emergent behaviour, considering the uncertainties associated with the behaviours of the individual components. Moreover, the emergent behaviour might be either beneficial or harmful. In the context of dependable systems, emergent behaviours that are harmful are as important as the beneficial behaviours since the former might lead to new failure behaviours that were not considered during system design. The incorporation of means that are able to clearly identify and promote either of these still remains a challenge. A possible solution would be to restrict the system-emergent behaviours by imposing safety margins that would guarantee that the system behaviour would not diverge from the intended one.

As an example of a system based on process description, let us consider a fault-tolerant system based on crash failure semantics and dynamic reconfiguration (de Lemos, 2005b). Crash failure is the simplest and most restrictive type of failure where a component never produces an incorrect output. This can be implemented by comparing two streams of data, while assuming that only one

https://doi.org/10.1017/S0269888906000853 Published online by Cambridge University Press

might fail.Whenever there is a discrepancy between the two channels, the component processing halts before an incorrect outcome is produced. On the other hand, dynamic reconfiguration can be employed for handling replication of resources in self-managed systems. 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.

As an example of a system based on data description, let us consider adaptive anomaly detection as the first stage for achieving self-management in the context of dependable embedded systems (Ayara *et al.,* 2005). If a system is to be placed in an environmentwhose features are not fully known during design time, or an environment that is expected to change, then the system should be able to identify new anomalies that might affect its operation. In a fully autonomous system, as the system learns from its operation, new anomalies are identified, and their respective detectors are incorporated into the system's repertoire of detectors. Within this context, trust can be placed on the ability on an individual system to identify new anomalies and create new detectors. However in the context of a network of such systems in which new detectors might be distributed among other similar systems, trust cannot be placed on this vaccination-like process because of the different environments in which these systems are placed. In this situation, trust on the new detectors can only be obtained if a domain expert validates them.

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, the provision of assurances concerning their predictability is still insufficient, mainly in the context of dependable applications. However, the degree of predictability should depend on how strong the role of ''self'' is when dealing with unexpected circumstances. If the objective is to build a self-managed system that is able to handle the unexpected, then this system might not be predictable as required, thus affecting how much trust can be placed upon it. On the other hand, if a system should be sufficiently robust for avoiding dealing autonomously with the unexpected, then the problem is how to engineer the complexity of such a system. A promising approach for obtaining trust from systems that are based on data description is to build massively redundant systems in which the failure of some of its components would not affect the expected outcome of the whole system. However, for such a solution to be successful, diverse data-oriented approaches should be composed in order to increase their combined effectiveness, or coverage. However, a major weakness in such configuration would be the quality of the training data. If the data are not good, it does not matter how many approaches are employed if all of them suffer the same deficiencies.

4 Trust-based electronic service communities

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. It is also useful to note that trust may not be a binary decision (i.e. whether a system is trustable or not) – but may be a continuous variable that reflects the ''degree'' of trust that one can place in a system. It is unlikely that in a distributed system one is likely to find an ''absolute'' measure of trust – as details about component behaviours and their current state cannot be known. It is therefore only possible to speak of ''relative'' trust – i.e. trust that can be estimated based on observed or predicted behaviour of the system as a whole.

Trust issues in autonomic computing may be classified according to the ''observer'' interacting with the system, and may include.

- Trust in the overall system behaviour: in this instance, an external user or system may wish to know if the autonomic system can produce a required output within certain constraints. The

https://doi.org/10.1017/S0269888906000853 Published online by Cambridge University Press

components that make up the autonomic system are not significant in this instance, as one is primarily concerned with the combined behaviour. Constraints on the result can include a time frame (i.e. whether a result is produced within 4 seconds, for instance), a particular type of result (i.e. up to 12 decimal point accuracy, for instance), etc.

- Trust in a ''provider'' component: in this instance, a component within the system may be interested in determining whether another component would provide a useful partner to interact with. Whereas the behaviour of the overall system may be difficult to model, it may be possible to assess trust for a component based on a model of the component. It is assumed here that the basis of the trust is the provider component offering some service to a client.
- Trust in a ''user'' component: in this instance, a component within the system may wish to determine whether it can trust another component to access its services. Generally, such an assessment has made use of security protocols in the past (to confirm the credentials of the client component).

A useful analysis of trust may involve an aggregation of concerns across these three levels. A useful question to address would be whether trust within a system is simply equivalent to combining trust in components (user and provider), or is trust in the combined behaviour of components more complex than such a simple combination? The location of the ''observer'' is also significant – as trust needs to be evaluated with reference to some ''context''. Context is dependent on the observer who is evaluating trust issues, but also more importantly on what aspect of trust is being evaluated. For instance, a service that is being evaluated on performance criteria (for instance, whether a service successfully delivers a certain performance profile based on its stated contract) should have a different trust rating on numerical accuracy. In order to undertake such an evaluation, it is therefore also necessary to specify some basis over which trust can be evaluated. For instance, if a service provider agrees to conform to a particular contract, then trust may be evaluated on the basis of whether such a contract has been successfully fulfilled in reality.

On the other hand, an observer may be a component within the autonomic system, in which case it would be capable of monitoring message exchanges between the components. Such an observer may be able to assess whether a particular component has fulfilled its stated contract with respect to a particular interaction protocol. Assessment in this instance is therefore based on the stated behaviour in the protocol description. Such an observer would be able to monitor the communication between components, but not their internal state. Alternatively, an observer may be the component itself, in which case it also has access to the internal state of the component, and is able to assess the trustworthiness of the component in fulfilling a particular set of obligations within the autonomic system, by observing the changes in internal state as a consequence of message exchanges and service execution.

A still more useful case would involve an observer who is outside the autonomic system – and only capable of assessing the behaviour of the system based on the result that the system produces. To be able to evaluate the trustworthiness of the system, it is now necessary to know the ''correct'' result as a response to a particular set of inputs and initial state. For instance, if a system is initialized from the reset state, and takes a set of inputs, it should produce a particular output. By assessing the correctness of this (previously known) output, it may be possible to place some degree of trust in the system. This is akin to the use of a ''learning'' data set to train a system (in machine learning) and subsequently evaluating its behaviour on a test set with known input/output combinations. However, it is not realistic to assume that an observer can constantly monitor a distributed system, and therefore the resulting trust that can be placed in such a system can only be an estimate.

Hence, in an autonomic system, 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. The concept of risk needs to be evaluated with reference to a particular context. For instance, risk may be associated with the likely disruption that may be caused to a client as a consequence of a provider violating

https://doi.org/10.1017/S0269888906000853 Published online by Cambridge University Press

pre-agreed constraints (for instance, not returning results within a particular time period). Overall, two main approaches may be deduced from literature. The first is based on allowing components 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. Such a trust model tries to capture the likely behaviour of another component, and uses this as a basis to test whether a ''correct'' result has been delivered. The second is based on allowing components within a system to calculate the amount of trust they can place in their interaction partners. This is achieved by guiding components in deciding on how, when and who to interact with. A component in this context refers to either a service user or a provider. However, in order to do so, trust models initially require components to gather some knowledge about their counterparts. This may be achieved in three ways.

- A presumption drawn from the component'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 (Witkowski *et al.,* 2001; Avizienis *et al.,* 2004). This aspect of trustmay 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. To be able to evaluate this level of trust, it is necessary for the client and service provider components to be supported with monitoring tools – such as Ganglia (Ganglia, 2005). The accuracy to which such monitoring tools can measure server information also impinges on the degree of trust that a client can place in a server component. It may also not be possible to directly monitor particular sets of parameters that are part of a trust model. A client component may therefore need to either derive such parameters based on the data that can be recorded, or else only make use of parameters that can be directly measured. Trust models therefore are constrained by the types of data that can be measured, derived or inferred by a component.
- Information gathered from other components: trust in this approach is computed indirectly from recommendations provided by others. As the recommendations could be unreliable, a component must be able to reason about the recommendations gathered from other components. The latter is achieved in different ways: (1) deploying rules to enable the component to decide to what degree recommendations provided by others can be trusted; (2) weighting the recommendation by the trust the agent has in the recommender component (Page *et al.,* 1999; Kamvar *et al.,* 2003). Such a referral mechanism may involve a multi-hop interaction. Retrieving such recommendations incurs a cost to the client component, and may limit the number of components that can be realistically asked. Determining the best set of other components to ask for recommendations is also a difficult issue, as the subsequent assessment of these recommendations incurs a cost. In recent research, there has been discussion of the use of a Trusted Third Party (TTP) as a means to acquire such recommendations. Such a TTP is responsible for combining feedback about the behaviour of components over a particular time period, and using this as a basis to calculate the trust that a client agent may place in that service providerwithin a particular context. Examples include the use of the feedback mechanism in electronic commerce systems such as eBay. The value of such a system is primarily limited by the motivation of clients to provide feedback for the benefit of other client components. There is also no general normalized scale which may be used to provide such feedback. Although a useful way to assess and compare trust between two service providers, this mechanism does not provide a useful way to assess trust that may be placed within one particular provider.
- Socio-Cognitive Trust: trust in this case is the capability to characterize the likely motivations of other components behaving in a particular way. This involves forming coherent beliefs about different characteristics of these components, and reasoning about these beliefs in order to decide how much trust should be put in them (de Limos, 2005b). Establishing a better understanding of motivations of other components may incur significant cost (in terms of computation and communication), and would also primarily be an estimate at best.

https://doi.org/10.1017/S0269888906000853 Published online by Cambridge University Press

A reason for acquiring trust is to allow client components to choose other partner components to interact with, to achieve a particular system objective. Generally, the basis of such ''partner selection'' is the functional capability of another component. For instance, a client component may wish to find a matrix solver component in the context of a scientific application. The choice of a component, in the context of Service Oriented Architecture (SOA), would involve querying a registry service to find suitable provider components. However, it is also possible for such a selection to be extended with trust information that has been acquired about a set of provider components. In this case, partner selection involves first evaluating the capability of a component, and then reducing the set of possible choices with trust information that may be held about provider components. 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 in 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. A component 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, a component would have to pay an initial cost to make some of these decisions. Subsequently, the component will only incur an ''operational'' cost – much lower than that for making some of these initial decisions. Formation of such communitiesmay be implicit, i.e. based on analysing interactions of components 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 components must be placed in a community. Communities, by definition are dynamic in nature, and will have a varying membership.We therefore suggest that an autonomic system within which component interaction is impacted by the trust components place in each other is a ''community''. As the process that led to the formation of such a community is based on trust between components, we suggest that such an autonomic system is also more trustworthy (i.e. likely to behave according to some predefined contract).

5 Integrating trust as autonomic intelligence

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. Contextawareness 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 smarthomes, 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 applicationspecific) 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 1–3m. Thus, we can define properties, which we call Quality of Context (QoC) attributes that characterize 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 intradependenceswithin the autonomic system's 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 (Huebscher & McCann, 2005).

https://doi.org/10.1017/S0269888906000853 Published online by Cambridge University Press

An application makes use of a number of context services that in turn use one or more context providers (CP). 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 greater or equal 0 mean the application is satisfied with this CP and values less than 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 CP's 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 will not 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 particular, the CPs are not necessarily reliable in the QoC that they advertise, yet our selection mechanism only correctly selects the best source if the QoC attributes of the CPs can be assumed to be correct. Thus, we need to take into account, in the selection process, how likely the CPs' advertised QoC is likely to be correct. To do this, we introduce the level of 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 10 cm is advertised, but the actual location is 50 cm from what is delivered by the CP, then the CP is being unreliable. Instead, a CP advertising a location precision of 100 cm but delivering information within 50 cm 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 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 crossvalidation with other CPs and feeds this feedback into a parameterized probability density function that is used to predict the CP's current trust. *Tw* should be estimated for every QoC attribute of a CP, although in practice this is difficult as insufficient feedback may be available, and thus is done only for the most important QoC attributes, e.g. precision and refresh rate. 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.

The reason for modelling *tw* as a probability distribution is because our trust estimate may only have a limited amount of certainty. Specifically, we start with *tw* modelled as a uniform distribution where every trust level is equally likely. Then, as more and more feedback is acquired, the distribution is biased towards themost likely level of trust based on the observed feedback.Hence, if there is only little feedback available to estimate a CP's *tw* (in a particular QoC), then the *tw* probability distribution will be weakly biased, i.e. there is no clear trust level that is much more likely than all others. On the other hand, if a lot of feedback is available, then the resulting probability distribution will be more strongly biased, and as a result we will have more confidence that the estimated *tw* is likely to be correct. Therefore, we not only estimate a CP's *tw*, but also the certainty we have in this estimate, which affects how much *tw* is going to play a role in the selection of the best alternative. This is necessary because taking into account a bad *tw* estimate may be more damaging than not taking it into account at all.

https://doi.org/10.1017/S0269888906000853 Published online by Cambridge University Press

Concluding, trustworthiness should be introduced as one of the metrics in the utility function deciding which alternative provider to pick, and we have found in an experimental case study that by introducing a trust level and using it in selecting the best alternative, 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 crossorganizationally 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).

References

Avizienis, A, Laprie, J-C, Randell B and Landwehr, C, 2004, Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* **1**(1) 11–33.

Ayara, M, Timmis, J, de Lemos, R and Forrest, S, 2005, Immunising automated teller machines. In *The 4th International Conference in Artificial Immune Systems (ICARIS 2005)*, Banff, Canada, August 2005, pp 404–417.

Castelfranchi, RFC, 1998, Principles of trust formas: cognitive anatomy, social importance and quantification. In *Proceedings of the International Conference on Multi-Agent Systems.*

Cristian, F, 1991, Understanding fault-tolerant distributed systems. *Communications of the ACM* **34**(2) 56–78.

de Lemos, R, 2005a, The conflict between self-* capabilities and predictability. In Babaoglu, O, Jelasity, M, Montresor, A, Fetzer, C, Leonardi, S, van Moorsel, A and van Steen, M (eds), *Self-Star Properties in Complex Information Systems (Lecture Notes in Computer Science, 3460).* Berlin: Springer, pp 219–229.

de Lemos, R, 2005b, Architecting web services applications for improving availability. In de Lemos, R, Gacek, C and Romanovsky, A, (eds), *Architecting Dependable Systems III (Lecture Notes in Computer Science, 3549).* Berlin: Springer.

Ganglia, 2005, Ganglia Monitoring System. Available at: `http://ganglia.sourceforge.net/`

Huebscher, MC and McCann, JA, 2005, A learning model for trustworthiness of context-awareness services. In *Proceedings of the 3rd International Conference on Pervasive Computing and Communications (PerCom).*

Kamvar, SD, Schlosser, MT and Garcia-Molina, H, 2003, The Eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th International World Wide Web Conference.*

MacFarlane, AGJ, 1993, Information, knowledge and control. In Trentelman, HL and Willems, JC (eds), *Essays on Control: Perspectives in the Theory and its Applications.* Boston: Birkhäuser, pp 1–28.

Page, L, Brin, S, Motwani, R and Winograd, T, 1999, The PageRank citation ranking: bringing order to the web. Technical Report SIDL-WP-1999–0120, Stanford Digital Library Technologies Project, Stanford University: `http://dbpubs.stanford.edu:8090/pub/1999-66`.

Parnas, D, 1972, On the criteria to be used in decomposing systems into modules. *Communications of the ACM***15**(12), 1053–1058.

Powell, D and Stroud, R (eds), 2003, *Conceptual Model and Architecture of MAFTIA.* MAFTIA deliverable D21. 1ST Project on Malicious- and Accidental-Fault Tolerance for Internet Applications (MAFTIA). IST-1999–11583, p 123.

Sabater, J and Sierra, C, 2002, REGRET: a reputation model for gregarious societies. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agents Systems.*

Simon, HA, 1981, *The Sciences of the Artificial*, 2nd edn Cambridge, MA, USA: MIT Press.

Witkowski, M, Aritikis, A and Pitt, J, 2001, Experiments in building experiential trust in a society of objective-trust based agents. In Falcone, R, Singh, M and Tan, Y-H (eds), *Trust in Cyber-societies (Lecture Notes in Computer Science, 2246).* Berlin: Springer, pp 111–132.

https://doi.org/10.1017/S0269888906000853 Published online by Cambridge University Press