INTERNATIONAL JOURNAL OF COMPUTERS
Issue 3, Volume 1, 2007

Digital Management of Trust for Component Software

Zheng Yan, Valtteri Niemi

***Abstract*—** Trust plays an important role in a software system, especially when the system is component based and varies due to component joining and leaving. How to manage trust in such a system is crucial for an embedded device, such as a mobile phone. This article introduces a trustworthy middleware architecture that can digitally manage trust in an autonomic way through adopting a number of algorithms for trust prediction, assessment and maintenance with regard to software component download and execution.

***Keywords*—**component software, trust, trusted computing, trust management, trust modeling.

I. Introduction

The growing importance of software introduces special requirements on trust. This normally implies that system software consists of a number of components that are combined to provide user features. Components interact over well defined interfaces; they are exported to applications that can combine and use the components. Thus, common components can be effectively shared by applications. A typical feature of devices with component software support is to allow addition of components after deployment, which creates the need for trust management with regard to software component download and execution.

In this article, we introduce a solution of autonomic trust management for a component software system. We aim to build up a trustworthy middleware architecture in order to support easy and late integration of software from multiple suppliers and still have dependable and secure operation in the resulting system.

We adopt a holistic notion of trust which includes several properties, such as security, availability and reliability, depending on the requirements of a trustor. Hence trust is defined as the assessment of a trustor on how well the observed behavior of a trustee meets the trustor's own standards for an intended purpose [1]. The behavior of the trustee can be reflected and thus measured by a number of the trustee's quality attributes. From this, the critical characteristics of trust can be summarized. It is both subjective and dynamic. Concretely, trust is different for each individual in a certain situation and, sensitive to change due to the influence of many factors.

Obviously, it does not suffice to require the trustor (e.g. most possibly a digital system user) to make a lot of trust related decisions because that would destroy any attempt at user friendliness. For example, the user may not be informed enough to make sound decisions. Thus, establishing trust is quite a complex task with many optional actions to take. Rather trust should be managed automatically following a high level policy established by the trustor. We call such trust management autonomic.

Autonomic trust management concerns trust management in an autonomic processing way with regard to evidence collection, trust evaluation, and trust (re-)establishment and control [5]. We need a proper mechanism to support autonomic trust management not only on trust establishment, but also on trust sustaining. This is important for a component software system that should support trustworthy downloading and executing of the software components. In this article, we develop a trustworthy middleware architecture that can manage trust in an autonomic way through adopting a number of algorithms for trust assessment and maintenance with regard to software component download and execution.

The rest of the article is organized as follows. Section 2 specifies the trust issues related to component software. Section 3 presents a middleware architecture for digital management of trust for component software. The concrete solution of autonomic trust management is reported in Section 4 and Section 5. In Section 6, we compare our work with some related work. Finally, conclusions and future work are presented in the last section.

II. Trust Issues about Component software

For the component-centered aspect we must consider trust at several decision points: at download time and during execution. At a component download time, we need to consider whether a software provider can be trusted to offer a component. Furthermore, we need to predict whether the component is trustworthy for installation. More necessarily, when the component is executed, we have to ensure it can cooperate well with other components and the system provides expected performance and quality. The trust relationship between system entities changes during the above procedure.

When discussing a component software system, the

Manuscript received October 1, 2006. Revised version received March 10, 2007. This work was supported in part by the EU ITEA Trust4All project.

Zheng Yan is with the Nokia Research Center, Helsinki, Finland (e-mail: zheng.z.yan@nokia.com).

Valtteri Niemi is with the Nokia Research Center, Helsinki, Finland (e-mail: valtteri.niemi@nokia.com).

execution of components in relation to other entities of the system needs to be taken into account. Even though the component is trustworthy in isolation, the new joined component could cause problems because it will share system resources with others. This may impact the trustworthiness of the whole system. Consequently, the system needs mechanisms to control its performance, and to ensure its trustworthiness even if internal and external environment changes. Additionally, some applications (e.g. a health care service) need special support for trust management because they have high priority requirements, whereas other applications (e.g. games), while exhibiting a similar functionality (e.g. a network connection) will not have the same priority. Therefore, system-level trustworthiness is dependent on the application domain. So the system needs a trust management framework that can support different trust requirements from the same or different software components, depending on the context they are used.

III. ARCHITECTURE

The architecture of the component software system consists of layered structure: an application layer that provides features to a user; a component-based middleware layer that provides functionality to applications; and, a platform layer that provides access to lower-level hardware. Using components to construct the middleware layer divides this layer into two sub-layers: a component sub-layer that contains a number of executable components and a runtime environment (RE) sub-layer that supports component deployment and execution.

The component runtime supporting frameworks also exist at the RE sub-layer. They provide functionalities for supporting component properties and for managing components. These frameworks also impose constraints on the components, with regard to mandatory interfaces, associated metadata etc. The runtime environment consists of a component framework that treats DLL (Dynamic Link Library)-like components. It provides a system-level management of the component configuration inside a device. Each component contains services that are executed and used by applications. The services have interactions with other services; they consume resources; and, they have metadata attached. The trust model of the software component is one kind of the metadata. It indicates required resources for providing specified performance, requirements for cooperation with other components, a trust priority level and composition rules for composing this model with other trust models [1].

Some frameworks in the runtime environment have to be supported with platform layer functionality. For example, for a resource framework, support for resource usage accounting and enforcement is required from the platform layer. In terms of trust management, the system needs to provide security mechanisms, such as access control, storage protection, secret key generation and encryption/decryption. In this case the security framework offers functionalities for the use of security mechanisms, provided by the platform layer, to requests raised by a trust management framework in order to establish and maintain a secure system. The platform layer could also provide trusted computing support on the upper layers [2].

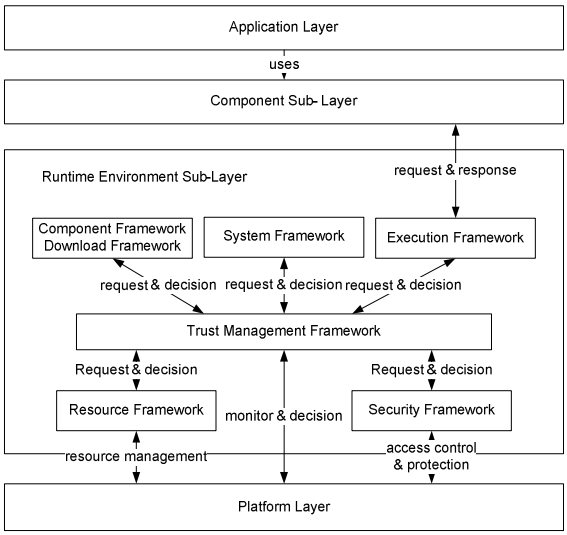


Fig. 1. Relationships among trust framework and other frameworks

Fig. 1 describes interactions among different functional blocks inside the runtime environment sub-layer. Placing trust management inside this architecture means linking the trust management framework with other frameworks responsible for the component management (including download), the security management, the system management and the resource management.

The trust management framework is responsible for the assessment on trust relationships and for automatically selecting suitable trust control mechanisms, system performance monitoring and autonomic trust management. The download framework requests the trust management framework to predict trust of components in order to decide if to download a component and which kind of mechanisms should be applied to this component. When a component service needs cooperation with other components' services, the execution framework will be involved, but the execution framework will firstly request the trust management framework to predict trust of the cooperation. Normally, multiple components with similar functionalities could exist or be available at the same time in the system. It is wise for the system to configure the components' cooperation based on trust prediction and assessment in order to achieve the best performance. Thereby, it is important for the system framework to configure the components according to the trust prediction or assessment results. Similarly, the trust management framework controls the security framework, to ensure that it applies the proper security mechanisms to maintain a trustworthy system. The trust management framework is located at the core of the runtime environment sub-layer. It monitors the system performance and instructs the resource framework to assign suitable resources to different processes. This allows the trust management framework to shut down any misbehaving component, and to gather evidence on the trustworthiness of a system entity. So briefly, the trust management framework acts like a critical system manager, ensuring that the system conforms to its trust policies.

IV. AUTONOMIC TRUST MANAGEMENT FOR COMPONENT SOFTWARE

As defined in [3], trust management is concerned with collecting the information required to make a trust relationship decision; evaluating the criteria related to the trust relationship as well as monitoring and re-evaluating existing trust relationships; and automating the process. We think that this concept needs to be extended in order to automatically control and ensure trust in a dynamically changed component software system. We employ autonomic trust management, which includes the following four aspects:

- Trust establishment: the process to establish a trust relationship between a trustor and a trustee.
- Trust monitoring: the trustor or its delegate monitors the behaviour of the trustee. The monitoring process aims to collect useful evidence for trust assessment.
- Trust assessment: the process for evaluating the trustworthiness of the trustee by the trustor or its delegate with respect to specified criteria or policy. The trustor assesses the current trust relationship and decides if this relationship has changed.
- Trust control and re-establishment: if the trust relationship has changed, the trustor will find reasons and make a decision if and which measures should be taken in order to control or re-establish the trust relationship.

A. Factors Related to Trust

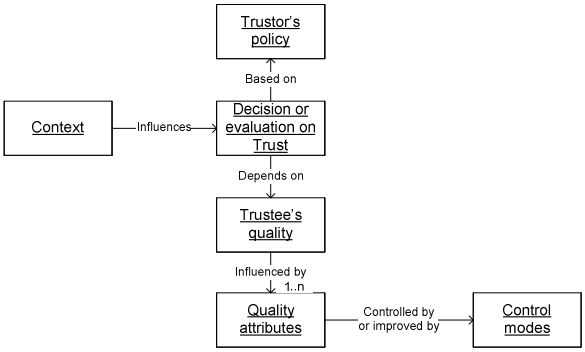


Fig. 2. Factors related to trust

We consider a component software system which is composed of a number of entities, e.g. a component (composition of components), an application, a sub-system and the whole system. The trustworthiness of an entity depends on a number of quality attributes of this entity. The quality attributes can be the entity's trust properties (e.g. security, availability and reliability) and recommendations or reputations with regard to this entity [16, 18]. The decision or assessment of trust is conducted based on the trustor's (e.g. a system user or his/her delegate) subjective criteria or policies and the trustee entity's quality attributes, as well as influenced by context. The context includes any information that can be used to characterize the situation of the involved entities. The quality attributes of the system entities can be controlled or improved by applying a number of control modes. Particularly, a control mode contains a number of control mechanisms or operations, e.g. encryption, authentication, hash code based integrity check, access control mechanisms, duplication of process, reconfiguration of component linkage, man-in-middle solution for improving availability, etc. It can be treated as a special configuration of trust management that can be provided by the system. The relationships of those factors related to the trustworthiness of a system entity are illustrated in Fig. 2.

B. A procedure of Autonomic Trust Management

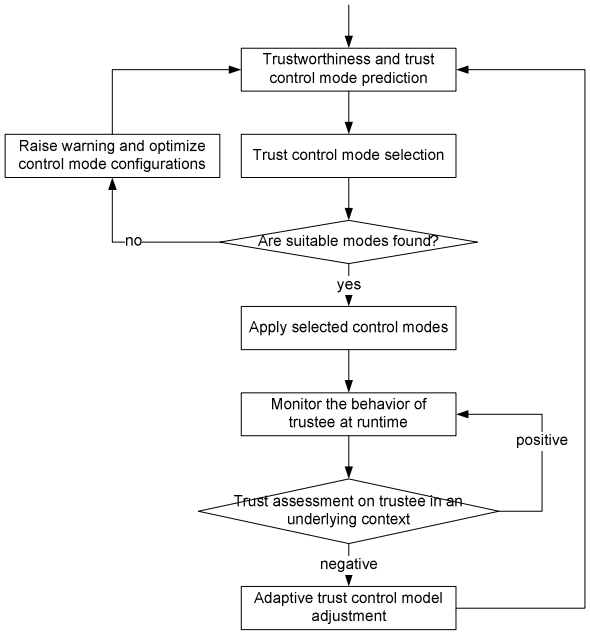


Fig. 3. An autonomic trust management procedure at runtime

Based on the above understanding, we propose a procedure to conduct autonomic trust management at runtime in the component software system targeting at a trustee entity specified by a trustor entity, as shown in Fig. 3.

Trust control mode prediction is a mechanism to anticipate the performance or feasibility of applying some control modes before taking a concrete action. It predicts the trust value supposed that some control modes are applied before the decision to initiate those modes is made. Trust control mode selection is a mechanism to select the most suitable trust control modes based on the prediction results.

For a registered trustor at the trust management framework, the trustworthiness of its specified trustee can be predicted regarding various control modes supported by the system. Based on the prediction results, a suitable set of control modes could be selected to establish the trust relationship between the trustor and the trustee. Further, a runtime trust assessment mechanism is triggered to evaluate the trustworthiness of the trustee through monitoring its behavior based on the instruction of the trustor's policies, as described in [1]. According to the runtime trust assessment results in the underlying context, the system conducts trust control model adjustment adaptively in order to reflect the real system situation if the assessed trustworthiness value is below an expected threshold. This threshold is generally set by the trustor to express its real expectation on the assessment. Then, the system repeats the procedure. The context-aware or situation-aware adaptability of the trust control model is

crucial to re-select suitable control modes in order to fulfill autonomic trust management.

C. A Trust Control Model

We developed a trust control model based on Fuzzy Cognitive Map [14] to support autonomic trust management [4, 5]. It is a signed directed graph with feedback, consisting of nodes and weighted arcs. Nodes of the graph are connected by signed and weighted arcs representing the causal relationships that exist between the nodes. As shown in Fig. 4, there are three layers of nodes in the graph. The node in the top layer is the trustworthiness of the system entity. The nodes located in the middle layer are the quality attributes of the entity, which have direct influence on the entity's trustworthiness. The nodes at the bottom layer are control modes that could be supported and applied inside the system. These control modes can control and thus improve the quality attributes. Therefore, they have indirect influence on the trustworthiness of the entity.

Concretely, a system entity's trustworthiness is influenced by a number of quality attributes $QA_i (i = 1,...,n)$. These quality attributes are ensured or controlled through a number of control modes supported by the system $C_j (j = 1,...,m)$. A control mode contains a number of control mechanisms or operations that can be provided by the system. We assume that the control modes are exclusive and that combinations of different modes are used.

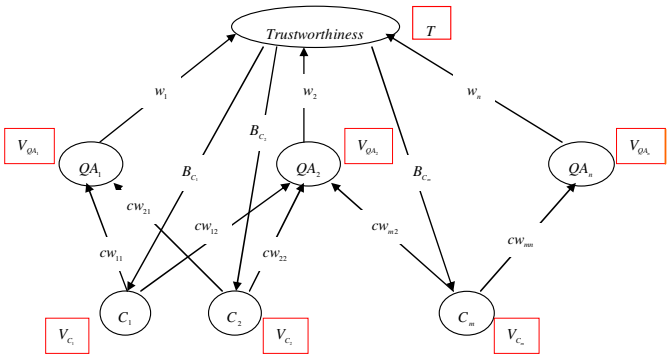


Fig. 4. Graphical modeling of trust control

Note that $V_{QA_i}, V_{C_j}, T \in [0,1]$, $w_i \in [0,1]$, and $cw_{ji} \in [-1,1]$. T^{old}, $V_{QA_i}^{old}$ and $V_{C_j}^{old}$ are old value of T, V_{QA_i}, and V_{C_j}, respectively. $\Delta T = T - T^{old}$ stands for the change of trustworthiness value. B_{C_j} reflects the current system configuration on which control modes are applied. The trustworthiness value T can be described as:

$$T = f\left(\sum_{i=1}^{n} w_i V_{QA_i} + T^{old}\right),$$

such that $\sum_{i=1}^{n} w_i = 1$. Where w_i is a weight that indicates the importance rate of the quality attribute QA_i regarding how much this quality attribute is considered at the trust decision or assessment. The weight w_i can be decided based on the trustor's criteria. We apply the Sigmoid function as a threshold function f: $f(x) = \frac{1}{1+e^{-\alpha x}}$ (e.g. $\alpha = 2$), to map node values V_{QA_i}, V_{C_j}, T into [0, 1]. The value of the quality attribute is denoted by V_{QA_i}. It can be calculated according to the following formula:

$$V_{QA_i} = f\left(\sum_{j=1}^{m} cw_{ji} V_{C_j} B_{C_j} + V_{QA_i}^{old}\right),$$

where cw_{ji} is the influence factor of control mode C_j on QA_i, cw_{ji} is set based on the impact of C_j on QA_i. Positive cw_{ji} means a positive influence of C_j on QA_i. Negative cw_{ji} implies a negative influence of C_j on QA_i. B_{Cj} is the selection factor of the control mode C_j, which can be either 1 if C_j is applied or 0 if C_j is not applied.

The value of the control mode can be calculated using $V_{C_j} = f\left(T \cdot B_{C_j} + V_{C_j}^{old}\right)$,

where T is the value of trustworthiness and B_{Cj} is the selection factor of the control mode C_j.

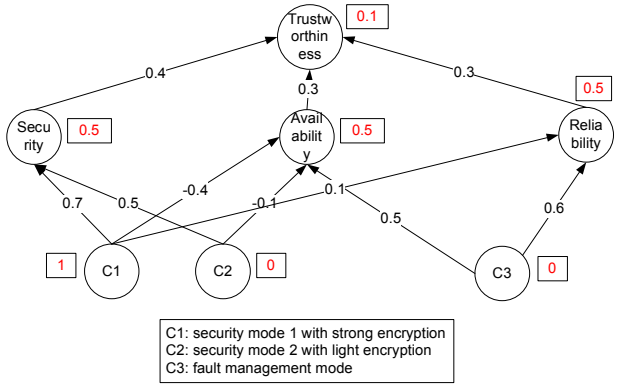


Fig. 5. An example of trust control model

An example of this model is shown in Fig. 5. The trustworthiness of the trustee entity is influenced by three quality attributes: QA_1 - Security; QA_2 - Availability; QA_3 - Reliability, with important rates $w_1 = 0.4$, $w_2 = 0.3$, and $w_3 = 0.3$, respectively. There are three control modes that could be provided by the system:

C_1: security mode 1 with a strong encryption service for encrypting data, but medium negative influence on availability.

C_2: security mode 2 with a light encryption service for encrypting data and light negative influence on availability.

C_3: fault management mode with positive improvement on availability and reliability.

The influence factors of each control mode to the quality attributes are specified by the arc weights. The values in the square boxes are initial values of the concept nodes. In practice, the initial value can be set as asserted one or expected one, which can be specified in the trustor's policy profile.

V. ALGORITHMS APPLIED FOR AUTONOMIC TRUST MANAGEMENT

There are a number of algorithms adopted by the trust management framework for autonomic trust management. For details, refer to [1, 4, 5, 12].

A. Trust Prediction for Component Software Downloading and Execution

Trustworthiness prediction is one of important issues that should be considered with regard to trust management of component software. The trustworthiness of a component should be predicted before initiating a concrete action, and this prediction should be comprehensive regarding multiple factors that could influence trust. We proposed two algorithms to predict trustworthiness for software components downloading and execution, respectively. The methodology is based on a trust model for component software, which indicates the component's asserted performance and requirements for achieving the performance. Through evaluating the related trust models and the component software system's competence, the algorithms can predict the trustworthiness of the software components. The prediction result is significant to determine whether to initiate the component downloading or start the execution of the component services. It also helps in locating system resources according to the trust priority level in case of any conflict [12]. Notably, the trust management framework applies this mechanism to cooperate with the download framework and the execution framework to aid establishing the trustworthiness of the system.

B. Trust Assessment at Runtime

We applied a simplified scheme of the Subjective Logic to conduct runtime trust assessment based on observation [1]. At runtime, a quality attribute monitor located at the trust management framework monitors the trustee's performance with respect to its quality attributes. For each quality attribute, if the monitored performance is better than the trustor's criteria the positive point of that attribute is increased by 1. If the monitored result is worse than the criteria, the negative point of that attribute is increased by 1. The trust opinion of each quality attribute can be generated based on the opinion generator θ [15]. In addition, based on the importance rates of different quality attributes, a combined opinion on the trustee can be calculated by applying the adding operator [1]. By comparing to a trust threshold opinion (*to*), the trust management framework can decide if the trustee is still trusted or not. The runtime trust assessment results play as the feedback to trigger trust control and re-establishment.

C. Control Mode Prediction and Selection

The trust control mode prediction is a mechanism to anticipate the performance or feasibility of some control modes supposed that those modes are applied before the decision to initiate them is made. We developed an algorithm based on the trust control model to conduct the trust control mode prediction as described in [4]. We further developed another algorithm in order to select the most suitable control modes based on the above prediction results. In the component software system, the control mode prediction and selection are important functionalities with regard to the automatic processing of trust management [5]. This mechanism also enables the trust management framework to optimize the underlying trust management configurations at runtime with regard to a trust relationship.

D. Adaptive Trust Control Model Adjustment

It is important for the trust control model to reflect the real system situation and context precisely. The influencing factors of each control mode should be context-aware. The trust control model should be dynamically maintained and optimized in order to reflect the real system situation. Thereby, it is sensitive to indicate the influence of each control mode on different quality attributes in a dynamically changed context. For example, when some malicious behaviors or attacks happen, the currently applied control modes can be found not feasible based on trust assessment. In this case, the influencing factors of the applied control modes should be adjusted in order to reflect the real system situation. Then, the system can automatically re-predict and re-select a set of new control modes in order to ensure the trustworthiness. In this way, the system can avoid using the attacked or useless trust control modes in an underlying context. Therefore, an adaptive trust control model is important for supporting autonomic trust management for the component software system. We developed a couple of schemes to adaptively adjust the trust control model in order to achieve the above purposes [5].

VI. RELATED WORK

A number of trusted computing and management work have been conducted in the literature and industry, which mostly focus on some specific aspects of trust. For example, TCG (Trusted Computing Group) aims to build up a trusted computing device on the basis of a secure hardware chip [2]. Some of trust management systems focus on protocols for establishing trust in a particular context, generally related to security requirements. Others make use of a trust policy language to allow the trustor to specify the criteria for a trustee to be considered trustworthy [3]. However, the focus on the security aspect of trust tends to assume that the other non-functional requirements [6], such as availability and reliability, have already been addressed. In addition, TCG based trusted computing solution can not handle the runtime trust management issues of component software.

Recently, many mechanisms and methodologies are developed for supporting trustworthy communications and collaborations among computing nodes in distributed systems [7-9]. These methodologies are based on digital modeling of trust for trust evaluation and management. However, most of existing solutions focus on the evaluation of trust, whilst they lack a proposal regarding how to manage trust based on the evaluation result. They generally ignore the influence of trust control mechanisms on trustworthiness. We found that these methods are not feasible for supporting the trustworthiness of a

device software system.

Regarding software engineering, trust has been recognized as an important factor for component software. A number of interesting solutions have been proposed to ensure its trustworthiness. Herrmann developed a special reputation system based on a component user's experience, other users' experiences and the third trusted party's certificate in order to reduce the expense of evaluating components [11, 16]. The runtime monitoring was implemented by a secure wrapper. It is a piece of code extending a component, while the wrapper does not change the behavior of the component. It monitors the component interface for security flaws. In addition, the intensity of the runtime observations about a component can be adjusted based on the current trust value of the component. Our work aim to conduct holistic trust management for component software based on the system's competence in an autonomic way. We apply a trust management framework at the component software RE sub-layer to conduct runtime observation based autonomic trust management in order to release the development burden and support interoperability. The trust assessment is based on observing a number of quality attributes of the trustee entity for the purpose of adaptively initiating trust control model adjustment to aware real system situation or context for autonomic trust management.

A framework for dynamic re-configuration of different qualities from the view of trust was constructed in [10, 17], which provides a common mechanism in middleware to ease the burden for trust component developers. Comparing with previous works, it focused on a trust perspective to satisfy various QoS demands of different users, and built a five-layer trust management framework, which not only provides common trust management facilities for trust components, but also supplies components for dynamical (re-)configuration of multi-properties. Based on the framework, the authors presented an algorithm to adjust dynamically all the involved trust properties according to predefined policies when the environment changes. The solution proposed in [10, 17] supports multiple properties of trust. The trust management is centralized in middleware, which is similar to our solution, but with different design since our design supports auto-selection of trust control mechanisms. Also, the trust evaluation function relies on users to customize. It is usually time-consuming and prone to errors. It needs some automation functions in the trust management framework to reduce more burdens of developers. Regarding the dynamic reconfiguration of component trust properties, it lacks necessary support to evaluate if trust can be managed based on the system's competence. The adjustment based on predefined policies lacks flexibility and can not predict cross-influence of various trust mechanisms on different trust properties. Our solution attempts to overcome the above problems and further release the burden of component software developers.

The on-going TrustSoft project aims to study a holistic approach to software trustworthiness through certifying multiple quality attributes of the software [13]. We argue that trust can be controlled according to its prediction or assessment result. Special control modes can be applied into the software system in order to ensure a trustworthy system in an autonomic approach.

VII. Conclusions

In this article, we summarized our results towards autonomic trust management for the component software system. Our main contributions include that we developed a couple of trust models to specify, predict, assess, set up and maintain the trust relationships that exist among system entities for the component software system. We further design an autonomic trust management architecture that adopts a number of algorithms for trust prediction, assessment and maintenance during component download and execution. These algorithms make use of the recent advances in Subjective Logic and Fuzzy Cognitive Map to ensure the management of trust within the component software system in an autonomic way.

For future work, we will further study the performance of the algorithms towards practical use of our results.

References

[1] Z. Yan and R. MacLaverty, "Autonomic trust management in a component based software system," in *Proc. ATC'06*, LNCS vol. 4158, pp. 279-292, 2006.

[2] Trusted Computing Group (TCG), TPM Specification, version 1.2, 2003. Available: https://www.trustedcomputinggroup.org/specs/TPM/

[3] T. Grandison and M. Sloman, "A survey of trust in internet applications," *IEEE Communications and Survey*, Forth Quarter, 3(4), pp. 2-16, 2000.

[4] Z. Yan, "A methodology to predict and select control modes for a trustworthy platform," *WSEAS Trans. on Computer*, Issue 3, vol. 6, pp 471-477, 2007.

[5] Z. Yan and C. Prehofer, "An adaptive trust control model for a trustworthy component software platform," in *Proc. ATC'07*, LNCS vol. 4610, pp. 226-238, 2007.

[6] S. Banerjee, C. A. Mattmann, N. Medvidovic, and L. Golubchik, "Leveraging architectural models to inject trust into software systems," *ACM SIGSOFT Software Engineering Notes*, in *Proc. the workshop on software engineering for secure systems—building trustworthy applications*, vol. 30, Issue 4, 2005.

[7] Z. Zhang, X. Wang, and Y. Wang, "A P2P global trust model based on recommendation," *in Proc. Int. Conf. on Machine Learning and Cybernetics*, vol. 7, pp. 3975-3980, 2005.

[8] C. Lin, V. Varadharajan, Y. Wang, and V. Pruthi, "Enhancing grid security with trust management," in *Proc. IEEE Int. Conf. on Services Computing (SCC 2004)*, pp. 303-310, 2004.

[9] Y. Sun, W. Yu, Z. Han, and K.J.R. Liu, "Information theoretic framework of trust modeling and evaluation for ad hoc networks," *IEEE J. Selected Area in Communications*, vol. 24, Issue 2, pp. 305-317, 2006.

[10] M. Zhou, H. Mei, and L. Zhang, "A multi-property trust model for reconfiguring component software," in *Proc. QAIC'05*, pp. 142-149, 2005.

[11] P. Herrmann, "Trust-based protection of software component users and designers," in *Proc. iTrust'03*, LNCS, vol. 2692, pp. 75-90, 2003.

[12] Z. Yan, "Predicting trustworthiness for component software," in *Proc. IEEE SecPerU'07*, pp. 1-6, 2007.

[13] W. Hasselbring, and R. Reussner, "Toward trustworthy software systems," *IEEE Computer*, vol. 39, Issue 4, pp. 91-92, 2006.

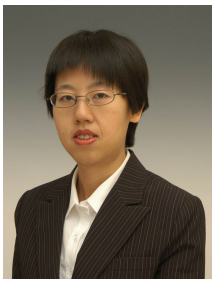
[14] B. Kosko, "Fuzzy cognitive maps," *Int. J. Man-Machine Studies*, vol. 24, pp. 65-75, 1986.

[15] A. Jøsang and S.J. Knapskog, "A metric for trusted systems," in *Proc. the 21st National Security Conference*, 1998.

[16] P. Herrmann, "Trust-based procurement support for software components," in *Proc. 4th Int. Conf. Electronic Commerce Research (ICECR04)*, pp. 505-514, 2001.

[17] M. Zhou, W. Jiao, and H. Mei, "Customizable framework for managing trusted components deployed on middleware," in *Proc. 10th IEEE Int. Conf. Engineering of Complex Computer Systems ICECCS*, pp. 283-291, 2005

[18] Z. Yan and S. Holtmanns, "Trust modeling and management: from social trust to digital trust," in *Computer Security, Privacy and Politics: Current Issues, Challenges and Solutions*, Subramanian R, Ed. IGI Global, 2008.



Zheng Yan received the B. Eng in electrical engineering and the M. Eng in computer science and engineering from the Xi'an Jiaotong University, Xi'an, China, in 1994 and 1997, respectively. She received the second M. Eng in information security from the National University of Singapore, Singapore, in 2000. She received the Licentiate of Science and the Doctor of Science in Technology in electrical engineering from the Helsinki University of Technology, Helsinki, Finland, in 2005 and 2007, respectively.

She is currently a Member of Research Staff at the Nokia Research Center, Helsinki, Finland. Before joining in the Nokia in 2000 as a Research Engineer and later on a Senior Research Engineer, she worked as a Research Scholar at the Institute for Information Research from 1997 to 1999 and a Software Engineer at the IBM partner SingaLab from 1999 to 2000, Singapore. She first-authored more than twenty paper publications and two book chapters. She is the inventor and co-inventor of eight patent applications. Her research interests are in trust modeling and management; trusted computing; mobile applications and services; usable security/trust, distributed systems and digital rights management.

Dr. Yan is a member of the IEEE and the IEEE Computer Society. She also serves as a program committee member for a number of international conferences and workshops.

Valtteri Niemi received the M.Sc degree from the University of Turku, Finland, Mathematics Department, in 1987 and the Ph. D. degree from the same department in 1989. He held several research and teaching posts at the University of Turku, including acting as an Associate Professor in Mathematics for the academic year 1992-3. In 1993, Niemi was nominated as an Associate Professor in the Mathematics and Statistics Department of the University of Vaasa, Finland, where he stayed until joining the Nokia Research Center, Mobile Networks laboratory, Helsinki in 1997. In 1999, he was nominated as a Research Fellow, and starting from 2004, he has been responsible for Nokia research in wireless security area.

Dr. Niemi now works on security issues in future mobile networks and terminals, the main emphasis being on cryptological aspects. He has participated 3GPP (3rd Generation Partnership Project) security standardization group from the beginning, and starting from 2003, he has been the chairman of the group.

In addition to cryptology and security, Dr. Niemi has done research on the area of formal languages. He has authored/co-authored around 40 scientific articles, he is a co-author of two books and more than 10 patents, and has frequently given talks in conferences and workshops.