Bridging Disjoint Trusted Domains into a Trustworthy System

ZHENG YAN
Nokia Research Center, Itämerenkatu 11-13, 00180 Helsinki, Finland

PENG ZHANG
Nokia Forum, Keilalahdentie 2-4, 00045, Espoo, Finland

Abstract: - With the rapid growth of digital computing and networking technologies, trust becomes an important aspect in the establishment of a secure digital system. Based on different reasons of trust, different trusted domains, possibly disjoint, are formed in a digital system, preventing the complete system from working improperly. What lacks therein are bridges that can link domains, across trust gaps to establish a complete trusted system. In this paper, we apply a methodology for establishing a trustworthy system through bridging disjoint domains of trust together. We illustrate how to analyze and design a trustworthy system by applying the methodology into a concrete example with regard to establishing a trustworthy middleware platform for component software.

Key-Words: - Trust management, Trusted computing, Security

1 Introduction

Trust is an important aspect in the establishment of a secure system [1]. However, trust is such a subjective and dynamic concept that different entities can hold different opinions on it even while facing the same situation [2]. Based on different trust perception, different trusted domains can be formed.

In today's digital systems, we can find many cases in which a system is actually formed by a number of the trusted domains and the communications and collaborations are actually conducted among and across those domains. A significant problem may arise from the fact that different domains must cooperate in order to provide a complete service even though they may not share the same concept of trust. Specifically, security problems may be caused by the deficiency of trust among domains. This deficiency is likely one of the major barriers that prevent the proliferation of digital communications and collaborations. The deficiency of trust is visible as gaps between the trusted domains established by different entities. For example, the proper selection of a number of component software to organize a trusted application (a trusted domain) has been causing a lot of problems [3].

Recently, many mechanisms and methodologies are developed for supporting trusted communications and collaborations among computing nodes in distributed systems (e.g. Ad Hoc Networks, P2P systems and GRID computing systems) [4-7]. These methodologies are based on digital modeling of trust for trust evaluation and management. Most of existing solutions focus on the evaluation of trust, but lack a proposal regarding how to manage trust based on the evaluation result. We found that these methods are feasible for supporting the trustworthiness of a digital system at the system runtime. However, little work considers a solution from the system analysis and design point of view.

Regarding trust modeling, various methodologies can be applied for solving different issues. Some trust models are based on sound technologies, e.g. PKI [8]. A big number of trust models are built up targeting at some special trust properties, such as reputations, recommendations and risk [9, 10]. Many trust models have been constructed for various computing paradigms such as GRID computing, ad hoc networks, and peer-to-peer systems [4-6]. In those models, some are computational, others are linguistic or graphic. Although a variety of trust models are available, it is still not well understood what fundamental criteria the trust models must follow. Without a good answer to this question, the design of trust models is still at the empirical stage [7].

In our previous work [11], we presented a methodology for bridging different disjoint trusted domains in mobile communications. In this paper, we apply the methodology and demonstrate it for establishing a trustworthy system by illustrating it with a real example.

The rest of the paper is organized as follows. Section two introduces the methodology to bridge the trusted domains in a digital system. In section three, we present the UML modeling of the methodology, and illustrate how to use it to analyze and design a trustworthy system through applying it into a concrete example in Section four. Finally, conclusions are provided in the last section.

2 Methodology to Bridge Different Domains of Trust

In this section, we introduce the methodology that helps to analyze trust inside any digital system by modeling the system into a number of trusted domains formed by different entities [11]. In order to solve the issue of trust gaps, we further propose three approaches to bridge the disjointed domains.

2.1 Definitions

Herein, we introduce some definitions that are related to the proposed methodology.

- **Trust**

We define *trust* as the confidence or belief of Entity A on another Entity B based on the expectation that Entity B will perform a particular action important to Entity A (trustor) [12]. The trustor could be a mobile device user, an enterprise company or a terminal node in an ad hoc network. The trustee could be a mobile device, a computing platform or a system providing various services.

- **Trusted domain**

The trusted domain is not an entirely new concept in the literature, but the following definition of the trusted domain is used herein. A *trusted domain* is a set of domain entities (e.g. service providers), defining trust statements and domain components (e.g. devices) such that all domain entities share certain trust statements regarding their trust definition for a specified purpose, and all domain components adhere to such trust definition and implement the statements. A trust statement identifies requirements of the domain entities to be trusted, and must be fulfilled by the domain components.

In Figure 1, an example of three trusted domains is presented. Domain *D1* consists of an entity *A* and two statements *s1* and *s2*. The statement *s1* does not define any existing component (i.e. there are no components that fulfill the statement) while the statement *s2* defines two components *a* and *b*. Domain *D2* contains entity *B* and two statements. The statement *s1* (identical with one of the statements from the domain *D1*) defines component *c* while statement *s3* defines component *d*. Finally domain *D3* has two entities: *C* and *D* with two statements. Statement *s4* defines two components *d* and *e*, in which the former is shared with the domain *D2*. Statement *s5* defines two components *e* and *f*, in which the former is also defined by the statement *s4*. Note that the component *d* fulfills both the statement *s3* and *s4*, so that the *D2* and *D3* are naturally bridged by the component *d*.

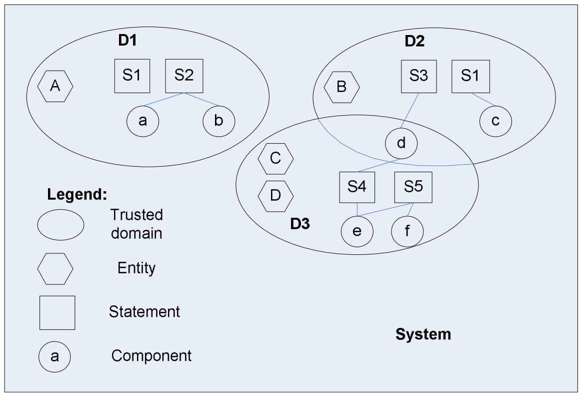

Figure 1: An example of three trusted domains in a system

In other words, a trusted domain is established whenever some entity or entities (such as a user, an operator or a service provider) trusts or trust some components for a specific purpose, regardless of the reasons for the trust that can be both subjective and objective, either rational or irrational. Herein, special interest is placed in the domains where their components are hardware or software components in a digital system.

- **Trusted bridge**

Based on the above definition of the trusted domain, we can see that full trust is retained inside the trusted domain while trust may be missing among the domains. This may cause a trust gap in places where the trusted domains do not overlap.

For the success of the digital systems, all the trusted domains that are essential for the complete system must intersect, i.e. there must be at least one component (or a chain of them) that is trusted by the entities communicating with each other. If it is not the case, a bridging solution should be identified and on that basis the bridging component must be created. Such a trusted bridge can be as simple as the component that is trusted by either domains, or complex, with its own respective entities, statements and components that can bridge the disjoint trusted domains.

A *trusted bridge* is a component or a set of components that is/are trusted by more than one domain. Therefore such component(s) can work as a bridge to establish trust among those domains. Note that if any of the bridged domains contain more than one statement, it is sufficient for the trusted bridge to implement one of those statements for each of the domains it is bridging.

In Figure 1, the domains *D1* and *D2* are disjoint and no trust can be readily established by any technical means, while the domains *D2* and *D3* intersect so that the component *d* can work as the trusted bridge to establish the trust between them.

2.2 Digital system organization

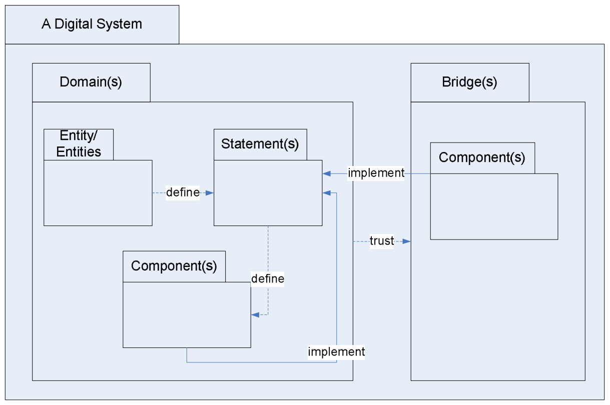

Figure 2: Digital communication system organization

In any digital communication system, we can always specify the system as a number of trusted domains. The communications and collaborations are actually conducted between those domains. Inside each trusted domain, the domain entities trust the domain components according to their defining statements, for whatever reasons they find appropriate. Among the trusted domains, it is expected that the trust must be usually created and constructed logically and rationally. We propose a methodology to analyze the trusted domains and to create the trusted bridge, effectively enabling the domains to form a complete solution as shown in Figure 2.

2.3 System modeling method

The proposed methodology is summarized as follows.

1. Model the digital system by separating it into a number of trusted domains formed by different entities.
2. Analyze each domain in order to extract the defining statements and list existing domain components. The resulting graph may resemble Figure 1.
3. For each pair of disjoint domains that must trust each other for the purpose of a given service, seek a bridging solution that can satisfy both domains (see the discussion below).
4. Form the trusted bridge by finding or creating a suitable component (or components), or by establishing bridging domains, depending on needs (see the discussion below).

This system modeling method can be illustrated in Figure 3 using a UML use case diagram.

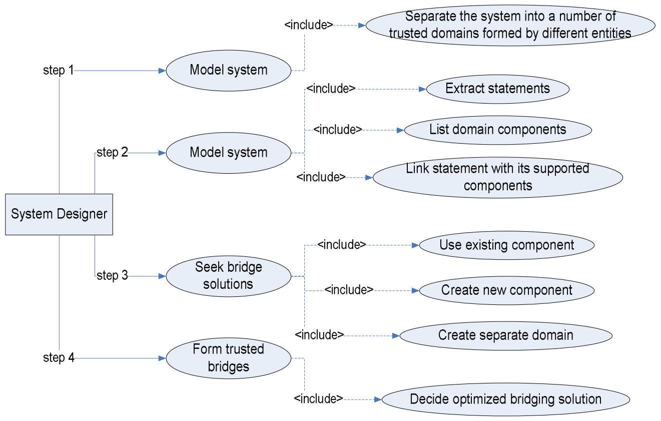

Figure 3: System modeling method

2.4 Trust bridging solutions

There are several approaches to identify the bridging solution and to introduce the trusted bridge, depending on the trust statements within the trusted domains as well as on non-technical limitations. Following is a short list of those. Throughout the discussion, the domains *D1* and *D2* from the definition of the trusted bridge (as shown in Figure 1) will be used to illustrate the defined concepts.

Approach A: Use an existing component (Figure 4.a.)

The analysis itself may lead to the discovery that there is already an existing component that may be trusted by more than one domain. Even though such a solution may seem trivial, it is the trust-based analysis itself that is frequently needed. Taking Figure 4.a as an example, as the domains *D1* and *D2* share the same defining statement *s1*, it is sufficient to verify that the component *c* (currently within the domain *D2*) that fulfills the statement *s1* is accepted also by *D1*.

Approach B: Create a new component (Figure 4.b.)

If the bridging component does not exist, it is possible to create it. Some components may conform to only one statement so that they require an identical statement in both domains. Some components may conform to more than one statement so that they can be used to bridge the domains with different statements. Note that the meaning of the identity of the trust statements requires further discussion that goes beyond the scope of this paper.

The use of a multi-statement component has been already demonstrated at the intersection of the domains *D2* and *D3* (as shown in Figure 1). Such a solution is also viable for the domains *D1* and *D2*, e.g. in the form of the component *g* that conforms to both the statements *s2* and *s1*, as shown in Figure 4.b.

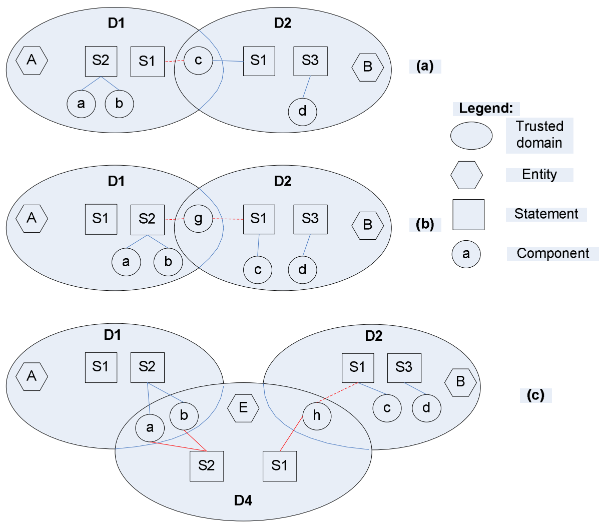

Figure 4: Methods of bridging trusted domains

Approach C: Create a separate domain (Figure 4.c.)

If there is no potential component that may satisfy the domains (e.g. the statements are significantly different), the solution may be to create a separate domain such that its domain components fulfill statements from both disjointed domains. Such a domain may share existing or new components with all the domains it is bridging. We call the created domain a bridging domain.

For example, domain *D4* can be introduced to bridge disjoint domains *D1* and *D2*. Domain *D4* consists of the entity *E* and three components: the existing components *a* and *b* that conform to the statement *s2* and is trusted also by the *D1* and a new component *h* that conforms to the statement *s1* and is trusted also by the *D2*.

If necessary, the creation of the new domain can be repeated to form a chain of domains until the bridging is complete, i.e. until there is at least one chain of domains that links all the domains that were originally disjoint. Obviously, it is possible to get multiple solutions to bridge trusted domains. It depends on further analysis and concrete systems requirements to decide which one is the best.

3 UML Modeling of the Methodology

It is significant that this methodology can be applied into any system analysis and design. It provides a special approach for security analysis from the trust point of view. Based on the analysis, people can define the topology of trust needed. It will be also potentially easier to find the proper component with appropriate technologies to bridge the trust gap that otherwise may cause security problems. Therefore, this methodology helps us to set up a secure and trusted system and aids us to seek new business opportunities, e.g. via seeking the proper trusted bridges to find new products or novel functions.

We further use UML (Unified Modeling Language) to make this methodology compatible with any digital system design. UML is a visual object oriented language for system and software development. It helps to present the system or application visually with graphical diagrams before actually programming or coding it. In this section, we use the UML to model the proposed methodology for supporting the design of a trustworthy system. In particular, we apply different UML diagrams in order to understand the methodology from different views.

The UML modeling includes several diagrams that present the concrete procedures of the methodology and the relationships among the system components. In addition, these diagrams virtually model the methodology from different views in order to clarify the relationships between the actor and the components.

3.1 Class diagram

The class diagram shown in Figure 5 further clarifies component relationships and basic attributes attached to each component. It also shows the methods that belong to each component object. What is more, this diagram works as the basis for the development of the methodology in order to support the design of a trustworthy digital system.

3.2 Collaboration diagram

The collaboration diagram shown in Figure 6 further identifies the runtime relationships among different design players targeting to work out a trustworthy digital system. This diagram helps on runtime view generation when applying this methodology.

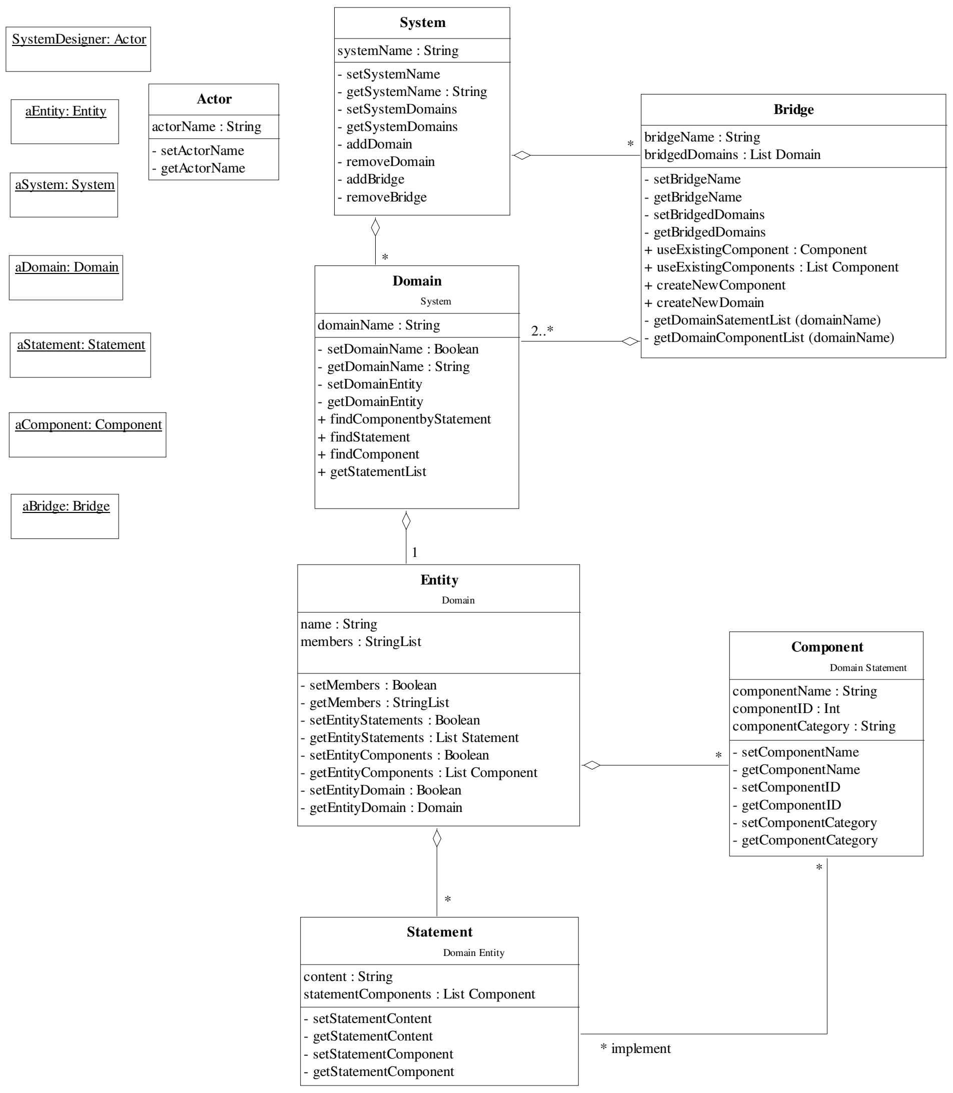

Figure 5: Class diagram of system components

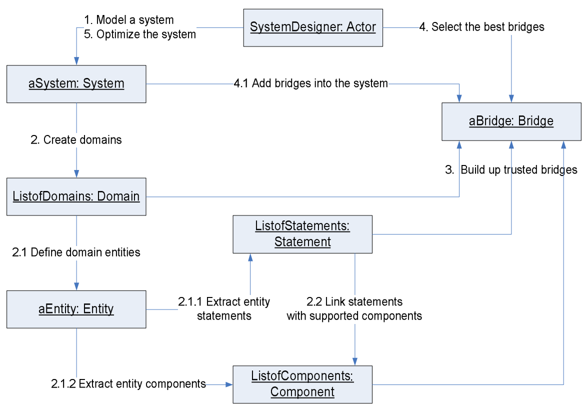
SystemDesigner: Actor
1. Model a system
5. Optimize the system
4. Select the best bridges
aSystem: System
4.1 Add bridges into the system
aBridge: Bridge
2. Create domains
3. Build up trusted bridges
ListofDomains: Domain
ListofStatements: Statement
2.1 Define domain entities
aEntity: Entity
2.1.1 Extract entity statements
2.2 Link statements with supported components
2.1.2 Extract entity components
ListofComponents: Component

Figure 6: Runtime relationships among system design players

4 Applicability Illustration

EU ITEA Trust4All project aims 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 of the resulting system [12]. This project is based on Robocop middleware runtime environment that has no any support on trust and security [13]. One important objective of Trust4All is to embed trust and security support into the existing Robocop component software platform. How to design a trustworthy system based on the existing Robocop platform is a vital issue of the Trust4All platform design.

The software architecture of the Robocop system consists of layered architecture: 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 divided this layer into two developmental layers: a component sub-layer that contains a number of executable components and a runtime environment (RE) sub-layer that supports component development.

The component runtime supporting frameworks exist at the runtime sub-layer. These frameworks provide functionalities for supporting component properties and for managing components. For example, a system framework takes care of system configurations related to the components. An execution framework will be involved when a component service needs cooperation with other components' services. A download framework is responsible for downloading a component. In addition, the runtime environment consists of a component framework that treats DLL-like components. This provides a system-level management of the software configuration inside a device. Each component contains services that are executed and used by applications. The services have interactions with other services and they consume resources. For some of the frameworks in the runtime environment, they have to be supported with platform functionality. For example, for a resource framework, support for resource usage accounting and enforcement is required from the platform layer.

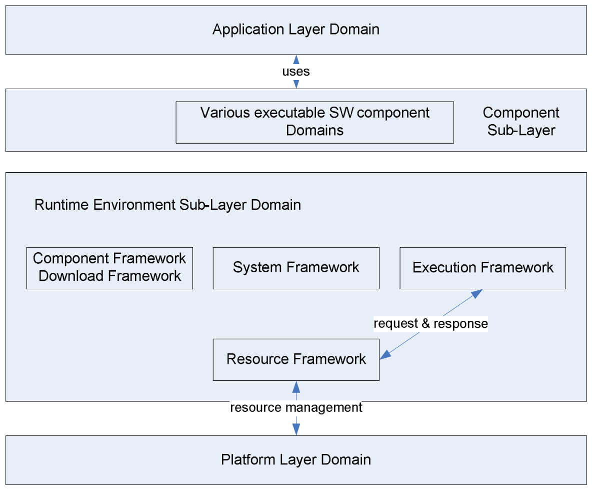

Figure 7: Trusted domains of Robocop system

The Robocop system can be modeled as a number of trusted domains based on the layered system architecture, as shown in Figure 7. Through making use of the methodology, we can create a number of trusted domains according to the existing Robocop design, as show in Table 1. It is understandable that there could be a number of executable software component domains in the system since the software components can be provided by different suppliers. Specifically, the application layer makes use of the services contained by various components in order to provide expected features to the device user.

As shown in Figure 7, there are three pairs of disjointed domains that must trust with each other for the purpose of providing a number of expected features to the device user: (a) the platform layer domain – the runtime environment sub-layer domain; (b) the runtime environment sub-layer domain – the various executable SW component domains; (c) the various executable SW component domain – the application layer domain.

Table 1: Detailed definitions of Robocop trusted domains

Name of Trusted Domain	**Domain Entity**	**Domain Components**	**Trust Statement**
Executable SW Component Domain	SW component provider or developer	Executable SW components that can cooperate in a trustworthy way	The provider or developer believes the SW components are in a good quality regarding provided services based on serious testing and commonly agreed interfaces.
Runtime environment sub-layer domain	Robocop designer	Component runtime supporting frameworks: component framework, download framework, system framework, execution framework, and resource framework	The Robocop designer believes this domain components provide functionalities for supporting SW component properties and for managing SW components. The runtime environment can be installed and executed at the device in a trustworthy way with the support of trusted computing.
Platform layer domain	Device vendor	Device hardware, OS, various resources (e.g. memory), SW and HW components related to trusted computing	The device vendor believes the device provides the support of trusted computing.

Application layer domain	Device user	Application user interface components	The device user believes that the interaction with the device user interface can achieve his/her expected features.

For the disjointed trusted domains (a), we firstly try Approach A and find a common trust statement “the support of trusted computing”, e.g. as described in [14]. That means the components related to the trusted computing that are implemented at the platform layer domain can ensure the trustworthiness of the upper layers [15, 16]. Therefore, those components themselves provide a trusted bridge for the platform layer domain and the runtime sub-layer domain.

Regarding the disjointed domains (b), we can not find any solution from Approach A. Then we try Approach B. We introduce a new component at the runtime environment layer – a trust management framework, as described in [15]. In order to build up the trust relationship among these domains, we applied both a ‘hard trust’ method and a ‘soft trust’ method. The ‘hard trust’ method uses the embedded trust management framework that plays as the trustor entity’s delegate to manage the trustworthiness of the trustee entity (e.g. a SW component). This trust management framework also supports applying a number of trust control mechanisms that can be used to ensure or sustain the trust relationships among the domain components. One important category of the trust control mechanisms is security related mechanisms, which include such mechanisms as encryption, decryption, access control mechanisms, authentication, hash code based integrity check, etc.

As for the ‘soft trust’ method, the trust assessment mechanism embedded in the trust management framework can assess the trustworthiness of a specified trustee entity based on runtime observation. In addition, the trust control prediction and selection mechanisms and the mechanisms for adaptive trust control model adjustment that are embedded in the trust management framework can further support and enhance autonomic trust management for the platform [17, 18]. Both methods fall into approach (b) – create a new component: the trust management framework to support autonomic trust management. In practice, this framework cooperates with other framework (e.g. the resource framework) to realize the whole system’s trust management. Thus, through introducing a new component, we generate a new trusted domain – Trust4All runtime environment sub-layer domain.

With regard to the third disjointed domains (c), we need a trusted bridge that can automatically manage various software components and make them cooperate together in a trustworthy way in order to offer expected services or features to the device user. Through using the methodology, we can not find any solutions from Approach A and B. We need to create a new domain as a trusted bridge to overcome the trust gap. Considering the newly created Trust4All runtime environment sub-layer domain, it can provide related trust support regarding component configuration, component execution, and communication protection among different software components through the cooperation of the trust management framework with other component runtime supporting frameworks. Thereby, the Trust4All runtime environment sub-layer domain can behave as a trusted bridge to bridge the various executable software component domains and the application layer domain together. The final design of the Trust4All middleware platform is shown in Figure 8 and depicted in details in [15].

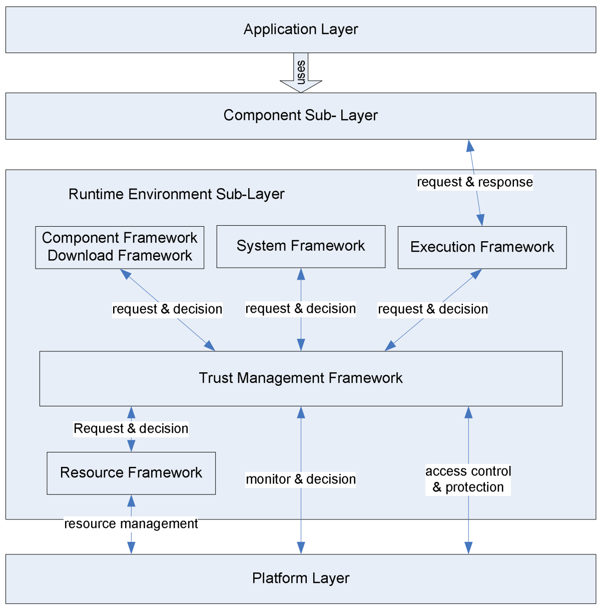

Figure 8: Trust4All system architecture

6 Conclusions

In this paper, we introduced and applied the presented methodology for establishing a trustworthy system. The methodology provides a trust modeling and analysis concept into the system architecture design. Based on the system analysis and trusted domain modeling, people can define the topology of trust in a digital system. This makes it potentially easier to find a proper component with appropriate technologies to bridge the trust gaps that otherwise may cause security problems. We illustrated how to use the methodology to analyze and design a trustworthy system through applying it into a concrete example with regard to establishing a trustworthy middleware platform for component software.

For future work, we will further refine the methodology in order to make it more practical and feasible for supporting various use cases. In addition, we will introduce some new features, such as evaluation and comparison of multiple trust bridge solutions and trust related feature binding and verification in software programming.

Acknowledgments
This work is partially sponsored by European Union ITEA Trust4All project. The authors would like to thank Dr. Piotr Cofta for his contributions to the previous paper about the methodology.

References:

[1] Diamadi, Z. Fischer, M.J.: A simple game for the study of trust in distributed systems. International Software Engineering Symposium 2001 (ISES'01), *Wuhan University Journal of Natural Sciences Conference* (March 2001).
[2] D. Gambetta.: Can We Trust Trust?. In *Trust: Making and breaking Cooperative Relations, Gambetta*, D (ed.) Basil Blackwell. Oxford, (1990).
[3] M. Zhou, H. Mei, L. Zhang, A Multi-Property Trust Model for Reconfiguring Component Software, *the Fifth International Conference on Quality Software QAIC2005*, 19-20 Sept. 2005,.
[4] Z. Zhang, X. Wang, Y. Wang, A P2P Global Trust Model Based on Recommendation, *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, Vol. 7, Aug. 2005
[5] G. Theodorakopoulos, J.S. Baras, On Trust Models and Trust Evaluation Metrics for Ad Hoc Networks, *IEEE Journal on Selected Areas in Communications*, Vol. 24, Issue 2, Feb. 2006
[6] C. Lin, V. Varadharajan, Y. Wang, V. Pruthi, Enhancing Grid Security with Trust Management**,** *Proceedings of IEEE International Conference on Services Computing* (SCC 2004), Sept. 2004
[7] Y. Sun, W. Yu, Z. Han, K.J.R. Liu, Information Theoretic Framework of Trust Modeling and Evaluation for Ad Hoc Networks, *IEEE Journal on Selected Area in Communications*, Vol. 24, Issue 2, Feb. 2006
[8] R. Perlman, An Overview of PKI Trust Models, *IEEE Network*, vol.13, no.6, Nov.-Dec. 1999
[9] L. Xiong, L. Liu, A Reputation-based Trust Model for Peer-to-Peer E-commerce Communities, *IEEE International Conference on E-Commerce*, CEC 2003
[10] Z. Liang, W. Shi, PET: A PErsonalized Trust Model with Reputation and Risk Evaluation for P2P Resource Sharing, *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, Jan. 2005
[11] Z. Yan and P. Cofta, "Methodology to Bridge Different Domains of Trust in Mobile Communications", In *Proceedings of the 1st International Conference on Trust Management (iTrust 2003)*, LNCS vol. 2692/2003, Greece, May 2003.
[12] Robocop, Space4U and Trust4All website: https://nlsvr2.ehv.campus.philips.com/
[13] J. Muskens and M. Chaudron, Integrity Management in Component Based Systems. In *Proceedings of the 30th EUROMICRO Conference* (Euromicro'04), Volume 00, Washington, DC, August 31 - September 03 2004, pp. 611-619.
[14] Trusted Computing Group (TCG), TPM Specification, version 1.2, 2003. https://www.trustedcomputinggroup.org/specs/TPM/
[15] Z. Yan, R. Maclaverty, "Autonomic Trust Management in a Component Based Software System", *the 3rd International Conference on Autonomic and Trusted Computing* (ATC06), LNCS vol. 4158/2006, China, September 2006.
[16] Z. Yan and P. Zhang, "A Trust Management System in Mobile Enterprise Networking", *WSEAS Transactions on Communications*, Issue 5, Vol. 5, May 2006.
[17] Z. Yan, "A Methodology to Predict and Select Control Modes for a Trustworthy Platform", *WSEAS Transactions on Computer*, Issue 3, Vol. 6, March 2007.
[18] Z. Yan, C. Prehofer, "An Adaptive Trust Control Model for a Trustworthy Component Software Platform ", *the 4th International Conference on Autonomic and Trusted Computing* (ATC07), LNCS, July 2007.