Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 179 (2007) 143–155
www.elsevier.com/locate/entcs

Managing Trustworthiness in Component-based Embedded Systems*

Gabriele Lenzini, Andrew Tokmakoff

Telematica Instituut, Brouwerijstraat 1,
7523 XC - Enschede, The Netherlands
{Gabriele.Lenzini,Andrew.Tokmakoff}@telin.nl

Johan Muskens

Philips Research, Prof. Holstlaan 4,
5656 AA - Eindhoven, The Netherlands
Johan.Muskens@philips.com

Abstract

Component-based systems use software components to achieve their overall high-level functionality which, in turn, may be extended by initiating the download of new components. This action may detrimentally affect the system's overall dependability and security characteristics. We address the problem of the enhancement of dependability and security for component-based embedded systems that run, for example, in consumer and embedded electronics devices. We propose a Trustworthiness Management Framework which, while acting on the behalf of components (Trustors), supervises the system's existing Trustor-Trustee relationships and preserves the overall system level of dependability and security. This is achieved by monitoring quality metrics on the components behaviours, by periodically evaluating their trustworthiness, and (when applicable) by controlling them. This paper focuses on the trustworthiness evaluation process offered by the Trustworthiness Management Framework. Trustworthiness evaluation is seen as a Trustors-parameterisable function. Trustworthiness is expressed with a triple of values: compliance, benignity and stability. The first measures the degree to which a component satisfies the Trustor's requirement; the second and third express the expected belief that, resp., the components will continue to be compliant and the component's behavioural qualities will remain stable. Trustworthiness is used by the Trustworthiness Manager Framework to make control decisions to regulate the system's overall dependability and security characteristics.

Keywords: component-based systems, trustworthiness evaluation, trustworthiness management architecture, dependability and security.

1 Introduction

Component-based software systems reach their high-level functionality through the use of a number of components which, in turn, provide a set of basic services. On

* This work has been conducted within the ITEA-EU project "Trust4All" which includes members from both industry and academia.

doi:10.1016/j.entcs.2006.08.038

a networked device, software components may also download new components into the system, thereby extending the system's capabilities at the risk of inadvertently hosting malicious or unstable components.

We address the problem of enhancing dependability and security of component-based embedded systems such as, for example, those supported by the Robocop/Space4U infrastructure [13]. This infrastructure runs on a diverse set of platforms for embedded consumer electronic devices, such as Symbian and Linux.

We study solutions that can be used to facilitate openness in such embedded systems without sacrificing their dependability and security characteristics. To this aim we introduce a Trustworthiness Management Framework (in short, TMF) which extends the existing Robocop/Space4U infrastructure with functionalities that allow it to evaluate (on the behalf of applications/Trustors) the trustworthiness of a component/Trustee with respect to some dependability and security quality requirements. The TMF is also responsible for looking after the satisfiability of the requirements of all the active Trustor-Trustee relationships in the system.

This paper focuses on the TMF's trustworthiness evaluation functionality. We express trustworthiness as a triple: compliance, benignity and stability. *Compliance* measures the degree of satisfiability of a component with respect to a set of Trustor-subjective quality requirements. *Benignity* and *stability* are defined as Subjective Logic [7] opinions; they express the Trustor-subjective expectation that a component's quality attributes will continue, respectively, to be compliant with the Trustor's requirements and to range over a neighbourhood of the values used for the last compliance evaluation.

Trustworthiness is used by the TMF to make control decisions. For example a negative compliance may result in action against the component e.g., its deactivation. A decrease in stability may trigger a re-estimation of the quality attributes and compliance re-evaluation, while a decrease in benignity may cause the component to be re-instantiated in a different mode of operation or in a controlled environment.

2 Related Work

Trust Management has recently attracted the attention of Computer Science research especial for the application areas of distributed access control and reputation networks management. (cf., [3,9]). The problem of trustworthiness evaluation that we address has many similarities with the task of selecting a trustworthy web service. Users generally favour web services that they expect will honour their agreements as described in the form of previously established a Service Level Agreement (SLA). For example [15] suggests that a web-service can be ranked according to a trust value that is calculated by a trusted registry using a set of user reports on the service over time.

A comparable approach, but in the domain of component-based systems, is discussed by Herrmann and Krumm in [5]; a component's trustworthiness is obtained from a Trust Information Service that stores components' reputation as experienced their users. The authors also propose a framework for security enhancement that

operate in a J2EE environment; here, components are controlled by Java-bean security wrappers that check the compliance of component's actual behaviour against its declared behaviour, encoded as state-dependent security constraints (expressed in terms of security automata or cTLA expressions). The level of the security checks depends on the component's trustworthiness, which wrappers obtain from the Trust Information Service (see also [4]).

Although our approach has some similarities the proposal of Herrmann and Krumm, the solution proposed be can not be effectively applied in the domain of resource-constrained embedded systems. Our components run in the Robocop/Space4U infrastructure which does not benefit from the security mechanisms offered by the Java Virtual Machine. In addition, we are interested in ensuring trustworthiness with respect to dependability and security qualitative characteristics, such as for example CPU consumption, memory usage, presence of encryption mechanisms, integrity data via checksums, mean time between failure, *et cetera*. Our TMF is embedded in the component middle-ware and acts with a resource management framework to take decisions on components at their instantiation time. Low level mechanisms, such as BSD jails or sandboxes, may be required in this case; alternatively, components can be re-configured to operate in a different mode of operation. An overall re-evaluation of the trustworthiness of each Trustor-Trustee relationship may be also reconsidered when a new component is admitted into the system, because this may impact on the overall resource distribution amongst components. Finally, our framework is designed to be parametrised, at run-time, by applications/Trustors that specify their subjective trustworthiness requirements in terms of metrics and a trustworthiness thresholds.

In [10], a trustworthiness measure is defined by two quantities: the ability to deliver the promised quality of service (called "conformance"), and the constancy in delivering the agreed level of quality (called "verity"), defined as the variance of the conformance over time. Our trust values benignity and stability are expressed as Subjective Logic opinions, and this facilitates their use in a recommendation management systems.

3 On Trust and Trustworthiness

The concept of "Trustworthiness" is related to the notion of "Trust". An entity is trustworthy (for a certain task) when we have a assurance that it will perform its promised service as expected [2]. Unfortunately, in many cases is almost impossible to have an unquestionable assurance about the trustworthiness of an entity; therefore, the choice of considering an entity trustworthy for a certain task involves a decision to trust. In this paper, we utilise the following definition of Trust [12]:

Definition 3.1 *Trust* is the extent to which one party (the Trustor) is willing to depend on another party (the Trustee) in a given situation with a feeling of relative security, even though negative consequences are possible.

Depending upon the specific scenario the Trustor and the Trustee roles can be

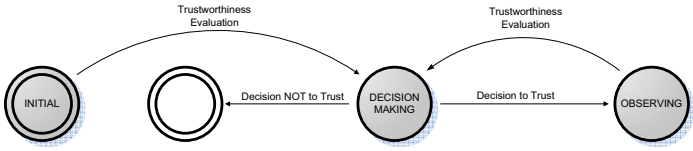

Fig. 1. States of a Trustor-Trustee relationship.

fulfilled by different types of entities. From the perspective of dependability and security, in this paper we consider Trustors and Trustees to be software components.

Example 3.2 *For her daily running exercises, Alice wears a T-shirt that has sensors and an open and component-based computing device attached. The device's main component, C, collects and stores data from the sensors. C may extend its functionality by acquiring new components, for example those in the class "Sweat-Heart". These components are able to communicate with a training centre that, in turn, analyses the heartbeat data and provides real-time notification to C if it detects any anomalies. In this case, the Trustor is the component C which is accountable for selecting and interacting with a "Sweat-Heart" component instance, whilst the Trustee is the specific "Sweat-Heart" component selected.*

4 Trustworthiness Manager Role

Figure 1 shows the states of a Trustor-Trustee relationship. Before making a decision to trust, a Trustor evaluates the trustworthiness of a Trustee. If the evaluation results in positive, the Trustor decides in favour of a Trustee. Later, analysis of observations on the Trustee's behaviour may trigger a re-evaluation of the trustworthiness and a new decision-making process.

When the Trustor is a software component in an embedded system, the "health" of Trustor-Trustee relationships may be assisted by a Trustworthiness Manager, which is the role played by the TMF (introduced in Section 1).

Before the *Trustworthiness Evaluation* transition in Figure 1, the Trustworthiness Manager, parametrised with the Trustor's requirements, can evaluate the trustworthiness of a specific Trustee on the behalf of the Trustor. We denote this functionality "Trustworthiness Evaluation Function" and it will be the topic of Section 8. Once the transition *Decision to Trust* has been made, the Trustworthiness Manager may perform observations regarding the Trustee's operation (e.g., performance statistics). Through analysis of this data, the Trustworthiness Manager is able to keep care of the Trustor-Trustee relationship (on the behalf of the Trustor) and to prevent potential damage that may occur when a Trustee starts failing to fulfil the Trustor's dependability and security requirements, that is, when its profile changes for the "worse"). In this case, the Trustor benefits, indirectly, from "damage control" mechanisms (e.g., BSD Jail [11], sandboxes [1], etc.) that the Trustworthiness Manager may utilise upon the Trustee.

These control mechanisms can also be activated by the Trustworthiness Manager in case of new Trustor-Trustee relationships within the system. For example, consider the situation in which a component has been admitted to the system, tak-

ing into account its assertion that it will never utilise more than 25% of the CPU in any given 1 minute period. If, for example, there is another component in the system that has been guaranteed 65% of the CPU in a given 1 minute period, and the newly added component is observed to be consuming 40% of the CPU, then the Trust Manager can either check if the first component is still compliant although using less CPU and in case not decide to "remove" the new component from the system as its behaviour threatens to de-stabilise the system as a whole.

In synthesis, the Trustworthiness Manager is able to ensure that the overall system remains in a "steady state" that respects all of the Trustor-Trustee relationships.

Example 4.1 (continued) *Alice decides to trial a version of the component "Sweat-Heart", on the condition that new component never discloses her data to third parties (e.g., the training centre) while she is training. Only when Alice is running in a competition, will she accept a small risk of information leakage against the benefit of an immediate feedbacks regarding possible heart dysfunctions. The Trustworthiness Manager, parametrised by component C, evaluates positively the component "Sweat-Heart" supplied by the Company "Fast&Well". This component can work in two modes, either "secure" or "fast". The "secure" mode implements secure (but slower and more power-consuming) communications (e.g., it ensures data integrity and confidentiality), whilst the "fast" mode utilises faster networking communications mechanisms without focusing on security aspects. After the new component has been admitted into the system, the Trustworthiness Manager continues to monitor its performances. When Alice is running in a competition and the communication speed of the component slows down, the Trustworthiness Manager (on the behalf of C) decide to switch the component to "fast" mode.*

5 Trustworthiness Management Framework

The TMF, developed using the Robocop/Space4U infrastructure [13], performs the role of Trust Manager as discussed in Section 4.

Moreover, the TMF makes use of "actuators" to convert the logical "control" decisions into actions which can, for example, reduce the priority of the component or signal it to change to a different mode of operation (with a difference set of asserted QA's) etc. For example, when a component is exhibiting QA metrics (through observation) that deviate from its expected QA's, it may be necessary to stop the component, or to perhaps re-start it. In this case, the Decision-making process makes a high-level decision (an action primitive) that is passed to the actuator(s). These actuators are responsible for mapping the decisions into concrete actions. This distinction separates decision-making from actions, where the one part is the "head" and the other part the "hands". Actuators may also be able to supply events to the decision-making process, forming a feedback loop. The in-depth discussion of the TMF's design, is ongoing work and beyond the scope of this paper.

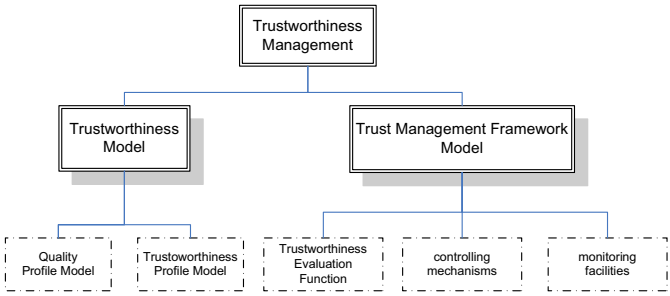

Fig. 2. Trustworthiness Management is defined in terms of the Trustworthiness Model and the Trust Management Model Framework.

6 Models for Trustworthiness Management

This section explains our approach in the design of the TMF. It based on the definition of two main models: the Trustworthiness Model and the Trust Management Framework Model, which are shown in Figure 2.

The *Trustworthiness Model* defines the structure of the information required for both Trustworthiness Evaluation and Trustor's decision-making. Our Trustworthiness Model suggests that the Trustee's profile is defined and described in a *Quality Profile Model*, and that the way of reasoning of the Trustor, when it uses a certain component, is supplied in the *Trustworthiness Profile Model*. We describe those models in Section 7.

The *Trust Management Framework Model* describes the mechanisms and solutions used for monitoring the Trustee's quality attributes metrics and for controlling certain Trustee quality attributes. Monitoring includes a mechanism that can obtain measurements e.g., CPU load, Process Memory footprint, network usage (number of sockets, bandwidth utilisation), containment level of a component, attempts to access unauthorised resources, component lock-up, component non-availability, etc. The control of quality attributes is realised, for example, by the possibility of switching the Trustee to operate in certain modes, or by the availability of controlled environments where the Trustee can be "safely" run.

Finally, the TMF Model specifies the *Trustworthiness Evaluation Function*, a functionality that is offered to the Trustor for evaluating a Trustee's trustworthiness (see Section 8).

7 Trustworthiness Model

The Trustworthiness Model is composed of a Quality Profile Model, associated with the Trustee and of a Trustworthiness Profile Model associated with the Trustor/Trustee relationship.

7.1 Quality Profile Model

The *Quality Profile Model* defines the information necessary to describe the behavioural characteristics of the Trustee. Formally, a Quality Profile Model *QP* is a

set $\{M_1, \ldots, M_k\}$ of modes, where each mode M is a set $\{QA_1, \ldots, QA_h\}$ of quality attributes, and each quality attribute QA is a tuple $\langle m_1, \ldots, m_l \rangle$ of metrics.

Components may have numerous modes of operation (e.g., "secure" mode, "fast" mode etc) , which may have different characteristics, called quality attributes (QA), that describe and quantify aspects of the component's behaviour e.g., in terms of its reliability, availability, etc. Each QA is specified by a defined tuple of metrics.

In [2], Avižienis *et al.* provided a taxonomy of the metrics that can be utilised for Dependability and Security. For example, the QA *Availability* (which is defined as "readiness for correct service"), can be expressed in terms of *Response Time* which expresses the average delay between the issuance of a service request and the moment in which the service is delivered, and *Uptime* which expresses the percentage of service requests that result in correct service provisioning. Instances of Quality Profiles (that we also write QA) carry the values of the metric and they can be easily coded into a XML file.

Example 7.1 (continued) *The Quality Profile Model of a "Sweat-Heart" component contains two modes, "secure" and "fast". Both of these modes are composed of an "availability" QA (in terms of "response time" in msec and "uptime" as a percentage), and a "security" QA (in terms of presence/absence of security features able to ensure confidentiality and integrity). An instance of the Quality Profile Model carries the behavioural information related to a specific component. For example, the following table*

availability		*security*	
response time	*uptime*	*confidentiality*	*integrity*
17	*0.98*	1	0

represents the QA of the "Fast&Well" component's "secure" mode (ref. Example 4.1). It indicates the "response time" is 17*msec, the "uptime" is, on average,* 98% *and that it implements mechanisms that ensure confidentiality but not integrity when communicating.*

7.2 Trustworthiness Profile Model

A *Trustworthiness Profile Model*, specified by a Trustor, is associated with each Trustor/Trustee relationship and it states: (i) the quality metrics the Trustor considers important for trustworthiness evaluation of a certain kind of component; (ii) the criteria to be used when evaluating the related metrics; (iii) the criteria to be adopted for local decision-making. Here, we only describe part of Trustworthiness Profile Model that is concerned with the trustworthiness evaluation function (i.e., items (i)-(ii)). This model has a similar structure to that of the Quality Profile Model except that its purpose is to represent the "demands" of the Trustor with respect to the Trustee.

A Trustworthiness Profile Model TP is a set $\{M_1, \ldots, M_{k'}\}$ of modes. As with the Quality Profile, the Trustworthiness Profile has modes that represent different modalities of judgements in different contexts. Each mode M is a set $\{(QA_1, W_1), \ldots, (QA_{h'}, W_{h'})\}$ of QA attributes and QA weightings. Weightings, such that

$\sum_{i=1}^{h'} W_i = 1$, expresses the importance the Trustor places upon specific quality attributes. Here, the Trustor can consider judgement only a subset of the characteristics specified in the Quality Profile Model of the potential Trustee. Each quality attribute QA is a tuple $\langle (m_1, \langle P_1, f_1^-, f_1^+ \rangle, w_1, \epsilon_1), \ldots, (m_l, \langle P_l, f_l^-, f_l^+ \rangle, w_l, \epsilon_l) \rangle$ of metrics, metric criteria, metric weightings and tolerance percentages. Metric weightings, such that $\sum_{i=1}^{l} w_i = 1$, express the importance the Trustor gives to each metric. The Trustor's criteria in judging the Trustee's quality attributes are formalised in terms of the following items:

1) a crisp predicate $P : T \to \{0; 1\}$.

Here T is the data domain of metric. It identifies the QA's metric values being either (crisply) compliant or not compliant from the Trustor's perspective;

2) two fuzzy sets $f^-, f^+ : T \to [0, 1]$.

Here f^- and f^+ measure the level of positive and negative compliance resp., of the metric with respect to the predicate. These fuzzy sets are defined via a pair of functions $c^+ : \{x : T, P(x)\} \to [0, 1]$ and $c^- : \{x : T, \neg P(x)\} \to [0, 1]$ (called *metric positive and metric negative compliance functions* resp.), as follows:

$$f^+(x) = \begin{cases} c^+(x), \text{ if } P(x) \\ 0, \text{ otherwise} \end{cases} \qquad f^-(x) = \begin{cases} c^-(x), \text{ if } \neg P(x) \\ 0, \text{ otherwise} \end{cases}$$

3) a tolerance percentage ϵ, which expresses the degree to which the Trustor will tolerate variation in the Trustee's observed metric value as absolute variations of ϵm.

Predicates, fuzzy sets, weightings and tolerance percentages are used to parameterise the Trustworthiness Evaluation Function, as explained in Section 8.

Example 7.2 (continued) *Alice's "training" mode Trustworthiness Profile expresses that the Trustee component should be assessed according to its availability and security QA's: the component's response time must be less than* 12*msec and its uptime at least* 95%*; moreover, it must exercise confidentiality mechanism, whilst the assurance of integrity is appreciated but optional. The following table summarises the associated criteria.*

	$W_{q_1} = 0.6$		$W_{q_2} = 0.4$	
	availability(q_1)		security(q_2)	
	$w_{x_1} = 0.5; \epsilon_{x_1} = 0.01$	$w_{x_2} = 0.5; \epsilon_{x_2} = 0.01$	$w_{y_1} = 0.75; \epsilon_{y_1} = 0$	$w_{y_2} = 0.25; \epsilon_{y_2} = 0$
	response time(x_1)	uptime(x_2)	confidentiality(y_1)	integrity(y_2)
P	$(x_1 < 12)$	$(x_2 > 0.95)$	$(y_1 = 1)$	$(y_2 <= 1)$
c^+	$1 - \frac{x_1}{12}$	$(\frac{x_2 - 0.95}{0.05})^2$	1	y_2
c^-	$1 - \frac{12}{x_1}$	$1 - \frac{x_2}{0.95}$	0	y_2

Here, availability is weighted 0.6, whilst security 0.4; this means that the former is more important to the Trustor than the latter. Within availability the two metrics are equally weighted, whilst within security, the presence of mechanisms for ensuring confidentiality is twice important as the presence of integrity mechanisms. Finally,

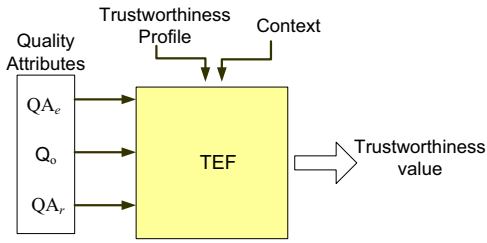

Fig. 3. The functional block of the Trustworthiness Evaluation Function.

the Trustor accepts a variance of 1% *in the measured availability while observing the "in service" behaviour of the Trustee .*

8 Trustworthiness Evaluation Function

The Trustworthiness Evaluation (TEF, in Figure 3) is conceived as a function $\texttt{tef} : \mathit{TProfiles} \times 2^{\mathit{QAttributes}} \times \mathit{Context} \rightarrow \mathit{TValues}$ that, given as input an instance of a Trustworthiness Profile Model, a set of QA (i.e., instances of the Trustee's Quality Profile Model), and contextual data, returns a trustworthiness value. Here *TProfiles*, *QAttributes*, *Context*, and *TValues* represent the data domains of the inputs and of the output respectively. We denote $\texttt{tef}_{(tp,c)}(\mathcal{Q})$ to be `tef` instantiated with the Trustor profile $tp \in \mathit{TProfiles}$ in the context $c \in \mathit{Context}$. Here, the context is managed in the easiest way i.e., to select the appropriate mode in *tp*; we refer to [16] for an advanced discussion about how to deal with context in trustworthiness evaluation. Therefore, $\texttt{tef}_{(tp,c)}$ transforms a set of QAs, $\mathcal{Q} \in 2^{\mathit{QAttributes}}$ into a subjective trustworthiness value that can be used for decision-making. Referring to Figure 3, we can see different QAs as inputs, viz. $\mathcal{Q} = QA_e \cup \mathcal{Q}_o \cup QA_r$. They correspond to different views on the QA's of the Trustee, namely:

- QA_e is the estimated QAs. It is the QA as understood by the Trustor. Initially (when the trustworthiness evaluation is called for before any decision to trust as shown in Figure 1) these are the QAs that the component (or someone on its behalf, e.g., the provider of the component) has "asserted". Further on, these are the QAs that the Trustworthiness Management Framework has estimated to better represent the "real" QAs of the Trustee.
- $\mathcal{Q}_o$ is the set of observed QAs. It is the set of QAs filled in by the TMF as the result of repeated monitoring activities. Initially this set is empty. Later, it denotes the set of observations collected by the TMF.
- QA_r is the set of recommendations that the Trustor obtains from its network of recommenders (e.g., see [8]). In this paper recommendations are discussed only marginally.

As a trustworthiness value $v \in \mathit{TValues}$, we propose a triple (CV, B, S). The CV, called *compliance*, is the value resulting from analysis of QA_e. It measures the degree of compliance of QA_e with respect to the Trustor's criteria (e.g., predicates, fuzzy sets and weightings); the B, called *benignity*, is the result of analysis of $\mathcal{Q}_o \cup QA_r$. It expresses the subjectiveinput expectation of the Trustor belief that a

component will continue to be compliant with its requirements; the S, called *stability* comes from the analysis of $\mathcal{Q}_o$. It expresses the Trustor's subjective belief that the current component's QAs will remain within the Trustor-specified tolerance percentage. The calculation of (CV, B, S) makes use of fuzzy sets [14] and Subjective Logic [7].

Fuzzy sets are used in the calculation of CV over the QA_e with the following algorithm: given a metric m of a $qa \in QA_e$ the corresponding fuzzy sets f^+, f^- in tp are used to calculate a single (w.r.t that metric) compliance as $cv(m) = f^+(m) - f^-(m)$. Here $cv : T \to [-1, 1]$, where T is the domain of the metric. An output in $(0, 1]$ shows a positive compliance (the degree of belonging to the compliant set), one in [-1,0) a negative compliance (the degree of belonging to the non-compliant set), while the value 0 expresses a neutral compliance. The CV over QA_e is calculated from single compliance values, as the following weighted summation that uses the weightings as specified in tp:

$$CV = \sum_{qa \in QA_e} W_{qa} \, (\sum_{m \in qa} w_m \;\; cv(m))$$

Example 8.1 (continued) *Let us consider the quality attributes specified in Example 7.1, and the trustworthiness profile given in Section 7.2 CV is calculated as follows*

$$\begin{aligned} CV = {} & 0.6\big(0.5\; cv(\textit{response time}) + 0.5\; cv(\textit{uptime})\big) + \\ & 0.4\big(0.75\; cv(\textit{confidentiality}) + 0.25\; cv(\textit{integrity})\big) \\ = {} & 0.6\big(0.5\; (-(1 - \frac{12}{17})) + 0.5\; (\frac{0.98 - 0.95}{0.95})^2\big) + 0.4\big(0.75\; (1) + 0.25\; (0)\big) \\ = {} & 0.319 \end{aligned}$$

that indicates that in general the requirements on the Trustee's QAs are fulfilled.

Subjective Logic (SL) [7] is used here to generate "opinions" on the subject "is the predicate P satisfied by $\mathcal{Q}_o$, the observed QA". A SL opinion is denoted by $\omega_P = (b_P, d_P, u_P)$. Here b_P, d_P, $u_P \in [0, 1]$ represent the subjective belief, disbelief and uncertainty respectively, in the truth of statement P, where $b_P + d_P + u_P = 1$. From an opinion ω_P we can obtain the probability, $E(\omega_P)$, of the expectation of the belief, by using the relation $E(\omega_P) = b_P + \frac{1}{2}u_P$. When the statement P, for example, says "the metric m is compliant" then ω_P can be interpreted as trust in the metric being compliant, whilst $E(\omega_P)$ is the probability of the expectation of the metric to be compliant [6]. Opinions can be combined using SL operators. Given two opinions ω_P and $\omega_{P'}$, over two predicates P and P', then $\omega_P \wedge \omega'_P$ is the opinion over the predicate $P \wedge P'$. Given two opinions ω_P^A and ω_P^B over P from different sources A and B, then $\omega_P^A \oplus \omega_P^B$ is the opinion that expresses a *consensus* e.g., a Bayesian fusion of the two opinions. The interested reader can find more detail on SL operators in [6,7].

Opinions can be generated from an *evidence space* [6]. The procedure is quite standard for the SL, with a slight personalisation to our case. Given an evidence space S and a predicate P, an opinion $\omega_P = (b, d, u)$ from S on P can be obtained

with the following formulae:

$$b = \frac{r}{r+s+t+1} \qquad d = \frac{s}{r+s+t+1} \qquad u = \frac{t+1}{r+s+t+1} \tag{1}$$

where r, s are the number of items in S that satisfy, do not satisfy resp., predicate P, whilst t is the number of items for which the evaluation of the predicate is uncertain (e.g., the datum was corrupted or missing, or we are using a three-valued predicate and the result cannot be interpreted as neither positive nor negative).

Benignity is the opinion $\omega_{P_{CV}}$ calculated using $\mathcal{Q}_o$ as evidence space. Here P_{CV} is the three-valued predicate that returns -1 if $CV < 0$, 0 if $CV = 0$, -1 otherwise. Informally, benignity models the Trustor's trust in the Trustee's behavior being compliant with its requirements. We can use benignity also when dealing with past Trustee's "reputation" according to the Trustor's viewpoint. A previous benignity value, B_{past}, can be combined with the present benignity, B, to obtain a (history-dependent) benignity $B' = B \oplus B_{\text{past}}$. Benignity value can also be used to recommend a Trustee to other Trustors. In this case, QA's in QA_r are benignity values experienced by other systems, and they can be combined with the local benignity to reach a final consensus. To deal with network of recommendations, and to cope with the related problems, can utilise the techniques developed in [8].

Stability is calculated as $S = \left(\bigwedge_{qa \in QA}(\bigwedge_{m \in qa} \omega_{P_{m,\epsilon}}\right)$. Here, $\omega_{P_{m,\epsilon}}$ is the opinion calculated using $\mathcal{Q}_o$ and the predicate $P_{m,\epsilon}$, which returns 1 if $m \in [m_e - \epsilon \cdot m_e, m_e + \epsilon \cdot m_e]$, -1 otherwise. Here, ϵ is the tolerance associated to m in tp, and m_e is the metric value used of the estimated QA. Stability models the Trustor's trust in the Trustee's behavior to remain in the neighbourhood of the estimated values, that is those used in the last compliance evaluation. Stability's can be made history-dependent if we combine the present stability with a past stability Trustee's "reputation" (according to the Trustor's viewpoint), viz. $S' = S \oplus S_{\text{past}}$.

Example 8.2 *In reference to Example 7.1 and Example 7.2, let us assume that from 100 observations of the "Sweat-Heart" component behaviour, 50 show a QA* $\{\langle 17, 0.98\rangle, \langle 1, 0\rangle\}$*, 50 a QA* $\{\langle 11, 0.96\rangle, \langle 1, 0\rangle\}$*, and one a QA* $\{\langle 12, 0.95\rangle, \langle 0, 1\rangle\}$*. The Benignity value we obtain, (0.98, 0.01, 0.01), shows a belief in the component to continue being compliant with the requirements (cf. Example 7.2). Stability results in a value (0.23, 0.75, 0.02) flagging that there is a disbelief in the stability of the overall QAs of the component, with respect to values used in the compliance evaluation. In this case, the presence of a disbelief means that it is better to re-estimate the* QA_e*. By using the same set of observations, we obtain the averaged* QA_e *being* $\{\langle 14.4, 0.96\rangle, \langle 0.5, 0\rangle\}$ *whose compliance is -0.338.*

9 Conclusion and Future Work

In this paper, we have introduced and discussed the development of a Trustworthiness Management Framework (TMF), which is a practical software solution used to enhance the dependability and security of networked, extensible, component-based embedded systems. The TMF, running in the middle-ware, supports components

(in the role of Trustor) to take the right decision to trust, and implements both solutions for monitoring the behaviour of a component and mechanisms of control that can be activated upon the component (in the role of Trustee) to safeguard the dependability and security.

We have focused on the Trustworthiness Evaluation Function (TEF) offered by the TMF, which components can parameterise according to the criteria they wish to use when evaluating the Trustee's compliance. Additionally, trustworthiness is evaluated in terms of: "benignity" (i.e., the belief that the Trustee will continue to be compliant) and "stability" (i.e., belief that the Trustee's behaver will remain within expected boundaries).

Benignity and stability are used for decision-making by the TMF, and the calculation of their values is defined in terms of Subjective Logic operators. The TEF has been prototyped in Ocaml [1] which has been interfaced with a Robocop [13] component written in C. We are now commencing a validation phase in which we will evaluate the use of the TMF in a variety of applications in the domains of health care, entertainment, and communication.

10 Acknowledgements

The work reported in this paper has been conducted within the Trust4All project. The authors would like to acknowledge the contribution of the project members in developing the concepts presented in this paper. We acknowledge Francisco Gómez Molinero, Arantxa Larrañaga Maiz, and Rube/n Alonso for the running example used through the paper. We thank Francisco Gómez Molinero, Hans Zandbelt, Zheng Yan and Arnoud Gouder and the anonymous reviewers for their useful comments.

References

[1] Acharya, A. and M. Raje, *Mapbox: Using parameterized behavior classes to confine untrusted applications*, in: *Proc. of the 9th USENIX Security Symposium Denver, Colorado, USA August 14 -17, 2000* (2000), pp. 1–18.

[2] Avižienis, A., J.-C. Laprie, B. Randell and C. Landwehr, *Basic concepts and taxonomy of dependable and secure computing*, IEEE, Transactions on Dependable And Secure Computing **1** (2004), pp. 11–34.

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

[4] Herrmann, P., *Trust-based protection of software component users and designers.*, in: P. Nixon and S. Terzis, editors, *Proc. of Trust Management, First International Conference on Trust Management (iTrust 2003), Heraklion, Crete, Greece, May 28-30, 2002*, LNCS **2692** (2003), pp. 75–90.

[5] Herrmann, P. and H. Krumm, *Trust-adapted enforcement of security policies in distributed component-structured applications.*, in: *Proc. of the Sixth IEEE Symposium on Computers and Communications (ISCC 2001), 3-5 July 2001, Hammamet, Tunisia* (2001), pp. 2–8.

[6] Jøsang, A., *A subjective metric of authentication*, in: J.-J. Quisquater, Y. Deswarte, C. Meadows and D. Gollmann, editors, *Proc. of the 5th European European Symposium on Research in Computer Security (ESORICS 98), Louvain-la-Neuve, Belgium, September 16-18, 1998, Proceedings*, LNCS **1485** (1998), pp. 329–344.

[1] http://caml.inria.fr

[7] Jøsang, A., *A logic for uncertain probabilities*, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **9** (2001), pp. 279–312.

[8] Jøsang, A., L. Gray and M. Kinateder, *Simplification and analysis of transitive trust networks*, Web Intelligence and Agent Systems Journal (2006), (to appear).

[9] Jøsang, A., R. Ismail and C. Boyd, *A survey of trust and reputation systems for online service provision*, Decision Support Systems (2005), (available on line on ScienceDirect) in press.

[10] Kalepu, S., S. Krishnaswamy and S. W. Loke, *Reputation = f(user ranking, compliance, verity)*, IEEE Tansactions on software engineering **30** (2004), pp. 311–327, (appeared also in Proc. of the IEEE International Conference on Web Services (ICWS'04)).

[11] Kamp, P. H. and R. N. M. Watson, *Jails: Confining the omnipotent root*, in: *in Proc. of the 2nd International Conference System Administration and Networking (SANE 2000) ,May 22 - 25, 2000, Maastricht, The Netherlands*, 2000, pp. 1–15.

[12] McKnight, D. H. and N. L. Chervany, *The meaning of trust*, Technical Report MISRC 96-04, University of Minnesota, Management Information Systems Research Center (1996).

[13] Muskens, J. and M. Chaudron, *Integrity management in component based systems*, in: *Proc. of the 30th EUROMICRO Conference, 1-3 Sept 2004, Rennes, France*, 2004.

[14] Nguyen, H. T. and E. A. Walker, "A First Course in Fuzzy Logic," Chapman & Hall/CRC, 2006, (Third Edition).

[15] Sherchan, W., S. W. Loke and S. Krishnaswamy, *A fuzzy model for reasoning about reputation in web services*, in: H. Haddad, editor, *Proc. of the 21st ACM Symposium on Applied Computing (ACM SAC 2006) - Trust, Recommendations, Evidence, and other Collaborative Know-how (TRECK), 23-27 April 2006, Lyon, France* (2006), pp. 1886–1892.

[16] Toivonen, S., G. Lenzini and I. Uusitalo, *Context-aware trust evaluation functions for dynamic reconfigurable systems*, in: *Proc. of the Models of Trust for the Web workshop (MTW'06), held with the 15th International World Wide Web Conference (WWW2006) May 22, 2006, Edinburgh, Scotland* (2006).