Supporting Coordinated Maintenance of System Trustworthiness and User Trust at Runtime

Torsten Bandyszak, Micha Moffie, Abigail Goldsteen, Panos Melas, Bassem Nasser, Costas Kalogiros, Gabriele Barni, Sandro Hartenstein, Giorgos Giotis, Thorsten Weyer

Supporting Coordinated Maintenance of System Trustworthiness and User Trust at Runtime

Torsten Bandyszak[1], Micha Moffie[2], Abigail Goldsteen[2], Panos Melas[3], Bassem I. Nasser[3], Costas Kalogiros[4], Gabriele Barni[5], Sandro Hartenstein[6], Giorgos Giotis[7], Thorsten Weyer[1]

[1]paluno - The Ruhr Institute for Software Technology, University of Duisburg-Essen, Germany
{torsten.bandyszak, thorsten.weyer}@paluno.uni-due.de

[2]IBM Research - Haifa, Israel
{moffie, abigailt}@il.ibm.com

[3]IT Innovation Centre, University of Southampton, UK
{pm, bmn}@it-innovation.soton.ac.uk

[4]Athens University of Economics and Business, Greece
ckalog@aueb.gr

[5]D-CIS Lab, Thales R&T, The Netherlands
Gabriele.Barni@D-CIS.NL

[6]Department of Economics, Brandenburg University of Applied Sciences, Germany
sandro.hartenstein@fh-brandenburg.de

[7]Athens Technology Center S.A., Athens, Greece
g.giotis@atc.gr

Abstract. In addition to design-time considerations, user trust and the trustworthiness of software-intensive socio-technical systems (STS) need to be maintained during runtime. Especially trust can only be monitored based on the actual usage of the system in operation. Service providers should be able to make informed decisions about runtime adaptation based on trust and trustworthiness, as well as respective essential relations. In this paper we present a unified approach to support the coordination of trust and trustworthiness maintenance. Trustworthiness maintenance is based on measuring objective system qualities, while trust maintenance considers two complementary measures of trust, i.e., the user behavior, and an estimation of the perceived system trustworthiness. A prototype tool demonstrates the feasibility of our approach. Furthermore, we illustrate specific functionalities of the tool by means of an application example.

Keywords: Trust · Trustworthiness · Run-Time Maintenance

1 Introduction

The success of software-intensive socio-technical systems (STS) increasingly depends on their users' trust in relevant system properties determining the system's trustworthiness. We consider trustworthiness of an STS its ability to fulfill the stakeholders' expectations (cf. [1]), which depends on a multitude of measureable software quality attributes, such as reliability or security [2,3]. In contrast, trust is a relationship between a person (trustor) and a system (cf. [1]). Trust involves subjectivity and uncertainty as it can be seen as a guess that the software will perform as expected (cf. [1]).

In order to assure the trustworthy operation of an STS and a high level of user trust in the system, it is not sufficient to consider these aspects during development. In particular, a user's trust can only be measured when the system is actually in operation. Service providers should thus be able to systematically assess both trust and trustworthiness at runtime, which requires early planning and installation of suitable monitoring sensors as well as actuators to invoke mitigations (cf. [4]). Furthermore, there is a reasonable interrelation between trust and trustworthiness, which only becomes visible at runtime. Besides the fact that trust is related to system trustworthiness, however, other factors influence the trust relationship as well. For instance, due to user experience, lack of transparency of the system's behavior, or the provider's reputation (cf. e.g. [1], [5]), the current subjective trust in the system may be low although objective system quality properties actually indicate a high degree of trustworthiness. Such a mismatch of trust and trustworthiness should be avoided [5]. Especially in complex STS, users may influence each other's trust as well. Hence, for making well-informed decisions on mitigation actions, it is crucial to consider comprehensive runtime information about the levels of both trust and trustworthiness.

Regarding trustworthiness maintenance, there are several approaches towards monitoring the quality of services (e.g. [6,7]). These approaches often only consider a few quality properties to be monitored, e.g., response time or availability. On the other hand, trust measurement and management approaches (e.g., [8,9]) deal with measuring the user behavior, or determining the user-perceived quality of specific services, most notably for real-time internet communication. However, these isolated approaches fail to address the challenges described above, i.e., to support making runtime decisions taking both trust and trustworthiness as well as the relationship between them into account. Hence, there is a lack of approaches that combine trust and trustworthiness aspects for runtime monitoring and management.

In this paper, we present our unified approach combining the maintenance of system trustworthiness and users' trust in STS. Our approach supports service administrators in coordinating the monitoring and management of trust and trustworthiness at runtime, as well as performing related mitigations. Trustworthiness maintenance is based on monitoring software services within an STS, and identifying threats caused by software properties (i.e., qualities or functions) not fulfilling the user expectations. We extend our preliminary results [10,11] with respect to trustworthiness maintenance, in order to consider trust and reflect the essential relationship between trust and trustworthiness as motivated above. Regarding trust maintenance, our approach combines two complementing ways of measuring users' trust: 1) estimating trust based on the different users' perceptions of the system's trustworthiness characteristics, and 2) monitoring the trust-related user behavior (e.g., in terms of number of mouse clicks in a certain time frame), which is heavily dependent on the type of STS.

Furthermore, we describe a tool prototype that implements our runtime trust and trustworthiness maintenance approach, and demonstrates its technical feasibility. The tool also allows validating our approach, and eliciting new requirements for extensions. As an initial evaluation of our approach, we applied the prototype to a case example of a secure web chat system. This application example involves three evaluation scenarios, each focusing on specific aspects of trust and trustworthiness maintenance. Details on the tool and the application example can be found in [12].

The remainder of this paper is organized as follows: Section 2 discusses related work. Section 3 provides an overview of our approach and sketches the conceptual solution. Section 4 presents a tool prototype that supports the application of our approach. In Section 5, we describe an application example that provides initial results of the ongoing evaluation of our approach. Finally, Section 6 concludes the paper.

2 Related Work

Regarding runtime monitoring and measurement of trustworthiness, an initial overview of related work can be found in our previous work [10]. Approaches for online monitoring of software services, e.g. [6], are based on observing the quality level of a service (QoS) with respect to guarantees defined in SLAs. These approaches usually cover only a limited set of quality properties such as availability or response time (cf., e.g. [7]). An overview of tools for monitoring QoS of cloud services is given in [13].

In [14], different service composition constructs and cost are taken into account for evaluating and managing the trustworthiness of a composite service-based system. A combination of quality properties monitoring and reputation is also possible (e.g. [7], [14]). The framework and trustworthiness evaluation method presented by Lenzini et al. [15] supports managing trust relationships, and aims at evaluating the trustworthiness of a trusted component with respect to the satisfaction of quality attributes and the expectation that these will remain satisfied and stable. For managing trustworthiness at runtime, the system composition can be adjusted, e.g., underperforming services or components may be substituted or restarted (cf., e.g. [15]). QoS-aware service selection also takes cost minimization objectives into account (cf., e.g. [16]).

Compared to user surveys, measuring the user behavior directly from the interactions with the system is a more promising approach for runtime trust maintenance. It is, however, challenging to define generic trust-related behavioral measures and metrics that can be used for runtime monitoring. Leichtenstern et al. [17] investigated the physiological behavior of website users by means of heart rate and eye tracking sensors to determine how to objectively measure trust-related behavior (attention and engagement). Regarding security, Blindspotter [18] is a user behavior monitor that aims at detecting abnormal user activities caused by e.g. hijacked accounts.

As mentioned in the introduction, the trust of users is also related to the perceived trustworthiness of the system, e.g., in terms of response times. In general, transparency of the system's trustworthiness characteristics helps achieve appropriate trust [5]. Studies such as [8,9] indicate a relation between the subjective quality of experience (QoE), i.e., the "overall acceptability of an application or service, as perceived subjectively by the end-user" [19], and the objective QoS, e.g., the user-perceived throughput on network level. This is particularly relevant for browsing and real-time web applications such as online gaming or VoIP (cf., e.g. [20]). A framework for measuring QoE of video conferencing, and controlling QoE in case of limited bandwidth is presented in [21]. For instance, QoE Monitor [22] and EvalVid [23] are free tools that support determining the perceived video conference transmission quality. Fiedler et al. [24] propose a generic formula to quantify relationships between QoE and network-level QoS, which aims at controlling QoE based on QoS monitoring.

For managing user trust, acceptable QoS characteristics of a system, e.g., the latency of web browsing, should be determined to allow for appropriate resource allocation [9]. To control QoE, additional parameters need to be considered as well. Zhang and Ansari [25] propose a framework for managing QoE that distinguishes a QoS/QoE reporting and a QoE management component to satisfy users' target QoS constraints.

As motivated in Section 1, evaluating the trustworthiness of a system together with the trust of its users is of vital importance for runtime maintenance. Some research effort has been spent to analyze the relationship between objective trustworthiness and subjective user trust. However, our state of the art analysis revealed a gap of approaches considering the combined runtime evaluation of both trust and trustworthiness to provide administrators comprehensive information for coordinating adaptation decisions. Furthermore, regarding complex, heterogeneous STS, there is a lack of approaches covering versatile quality characteristics, and different, complementary means to assess user trust. Available tools are also mostly based on narrow measurement concepts, or focus on specific applications. Sections 3 and 4 will introduce our conceptual solution and our prototype tool addressing these particular research gaps.

3 Coordinated Trust and Trustworthiness Maintenance

Fig. 1 gives an overview of our overall conceptual approach, which supports administrators in coordinating runtime maintenance of trustworthiness and trust within an STS. The monitor observes the behavior of the STS constituents (including software, humans, etc.), and reports respective misbehavior alerts to the management, which then determines appropriate controls to be executed by the mitigation. Since the mitigation is rather a technical issue, this paper focuses on monitoring and management.

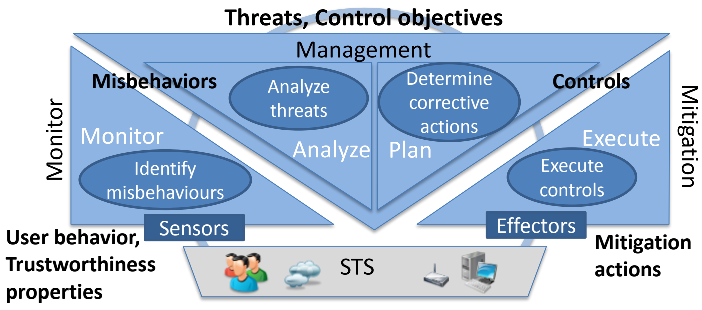


Fig. 1. Overview of our approach (based on [10])

To monitor trustworthiness at runtime, related functional and quality properties, such as response times, are reported by STS-specific sensors in the form of events. Based on these events, trustworthiness metrics are calculated to identify violations of user expectations in terms of the demanded quality of the software, which may be specified in SLAs for relevant quality criteria. Violations of specified values indicate the presence of threats (cf. [26]), which keep the system from performing as expected.

Concerning the monitoring of user trust in an STS, our approach subsumes the following two complementary approaches to support trust management decisions:

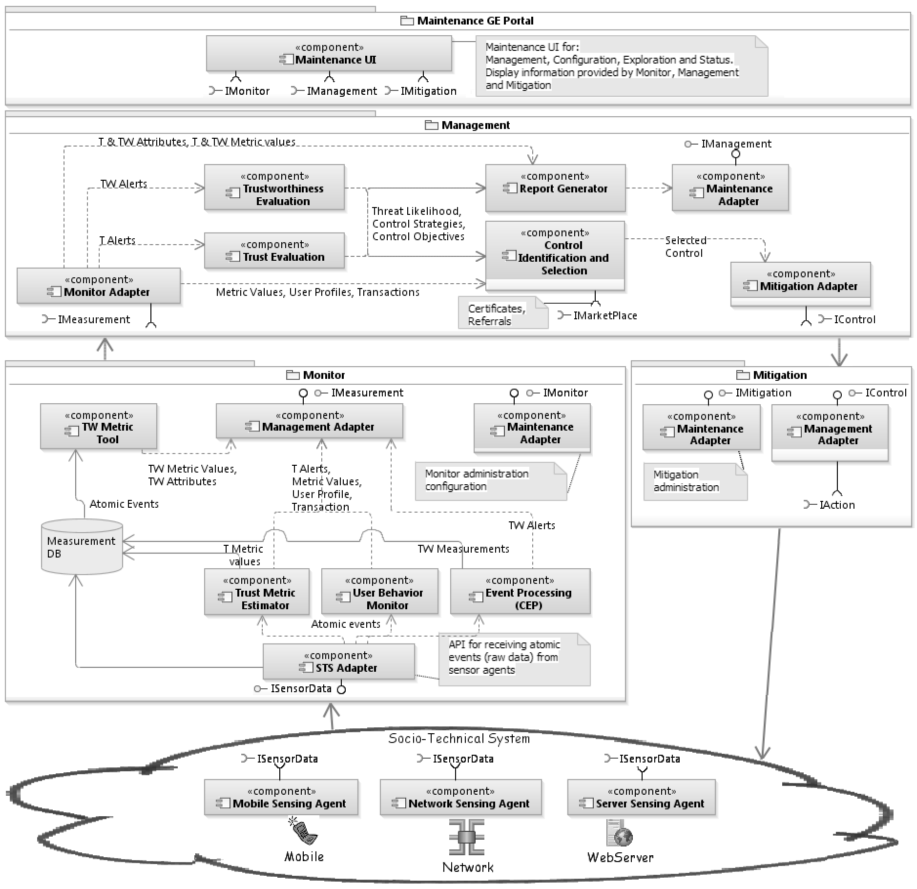
- The current level and evolution of a user's trust can be estimated at runtime based on the user's perception of the system's trustworthiness, which is characterized by respective metrics. This is based on the premise that each user is classified into one of four groups or segments, according to expected trust-related behaviors and relevant trustor attributes. Identifying the segment a user belongs to is done by contrasting the answers to be given in a predefined questionnaire before using the system to those already collected during a training period. An initial trust value and the update coefficients have been calculated for each of the four segments, and validated based on survey research. For example, members of the 'High Trust' group were found to consistently overestimate trustworthiness. For more details on user segmentation and trust level computation please see [27]. Calculating trust levels and selecting corresponding controls is based on the assumption that users use the system for a certain period (called optimization window). Based on their trust level at the end of each period, they decide whether to keep using the system, or not.
- Trust is also monitored and maintained by analyzing the user-specific, trust-related behavior. This approach considers each user separately, and requires respective STS sensors to report user-specific behavioral information. For instance, the number of a user's questions raised in a certain time interval can be considered a valuable source of trust-related information (see application example in Section 5).

Similar to trustworthiness maintenance (cf. [10,11]), both of these trust monitoring approaches are used to identify threats that are related to user trust. If a decrease in trust is detected based on either the estimated trust level indicating the user experience of trustworthiness, or abnormal user behavior when interacting with the system, an alert is issued to trigger the trust management process to analyze potential threats. Respective thresholds are defined for each of the user segments (cf. [27]). The management then analyses the likelihood of threats activity using semantic reasoning.

In case of any active threat to trust or trustworthiness, suitable controls are identified and selected based on a cost/benefit analysis (see [28] for more details on selecting an optimal control). The controls are then applied by executing mitigation actions on the STS. A control could be applied automatically (e.g., shutting down or substituting an underperforming service), or chosen and carried out by the administrator (e.g., contacting a specific user). Applying controls to restore trustworthiness will also reflect in the trust levels of the users. However, the relation between trust and trustworthiness also depends on other factors, which may, for instance, only be visible through monitoring the user behavior. By considering both trust and trustworthiness, our unified approach supports administrators in identifying and coordinating reasonable relationships between trust and trustworthiness, and their evolution during runtime.

4 Tool Prototype

Based on [12], this section presents the tool prototype supporting our approach. It will present the tool's architecture as well as the major components, and the user interface.



Maintenance GE Portal
«component» Maintenance UI
IMonitor
IManagement
IMitigation
Maintenance UI for: Management, Configuration, Exploration and Status. Display information provided by Monitor, Management and Mitigation
Management
T & TW Attributes, T & TW Metric values
IManagement
TW Alerts
«component» Trustworthiness Evaluation
«component» Report Generator
«component» Maintenance Adapter
Threat Likelihood, Control Strategies, Control Objectives
T Alerts
«component» Trust Evaluation
«component» Control Identification and Selection
Selected Control
«component» Monitor Adapter
«component» Mitigation Adapter
Metric Values, User Profiles, Transactions
Certificates, Referrals
IMarketPlace
IMeasurement
IControl
Monitor
IMeasurement
IMonitor
«component» TW Metric Tool
«component» Management Adapter
«component» Maintenance Adapter
TW Metric Values, TW Attributes
T Alerts, Metric Values, User Profile, Transaction
Monitor administration configuration
Atomic Events
Measurement DB
T Metric values
TW Measurements
TW Alerts
«component» Trust Metric Estimator
«component» User Behavior Monitor
«component» Event Processing (CEP)
Atomic events
API for receiving atomic events (raw data) from sensor agents
«component» STS Adapter
ISensorData
Mitigation
IMitigation
IControl
«component» Maintenance Adapter
«component» Management Adapter
Mitigation administration
IAction
Socio-Technical System
ISensorData
ISensorData
ISensorData
«component» Mobile Sensing Agent
«component» Network Sensing Agent
«component» Server Sensing Agent
Mobile
Network
WebServer

Fig. 2. Architecture of the trust and trustworthiness maintenance tool [12]

4.1 Tool Architecture

The initial tool architecture was presented in [10], including the overall tasks of the three main components Monitor, Management and Mitigation. This architecture was further refined in [11], describing the components involved in maintaining trustworthiness (TW). Fig. 2 shows the final tool architecture, including new components for trust (T) monitoring and management, as well as an optimal control selector, and a user interface for configuring, managing and viewing the trust and trustworthiness status of the system. The subsequent sections will describe the main components.

The *Monitor* is connected to the STS via system-specific sensors that report trust- and trustworthiness-relevant runtime monitoring data (cf. [4]). The *STS Adapter* forwards all atomic events received from the STS to three components in charge of the initial processing of these events and generating alerts upon deviation from normal (TW) or desired (T) behavior. The *Complex Event Processing* (CEP) is in charge of system-wide trustworthiness-related events, the *Trust Metric Estimator* (TME) estimates system trustworthiness as perceived by its different users, and the *User Behaviour Monitor* (UBM) collects data from user-specific sensors and estimates the trust each user has in the system. The *Monitor* also saves all atomic events in a *Measurement Database*, to be used in the GUI *Maintenance Portal* or for generating reports.

Both trustworthiness and trust alerts generated in the *Monitor* are forwarded to the *Management* component, where trustworthiness alerts are further processed by the *Trustworthiness Evaluator* (TWE), and trust alerts are further processed by the *Trust Evaluator* (TE). Both of these components generate a list of potential threats (including their likelihoods), and possible control strategies and objectives for mitigating these threats. The *Control Identification and Selection* component, using the *Optimal Control Selector* (OCS) sub-component, then suggests the most cost-effective control to be selected and deployed by the *Mitigation*. Respective feedback on the deployment of a control (i.e., the execution of a concrete mitigation action) is fed back to the respective *Management* components so that they can update their internal state accordingly. All detected threats, control strategies and deployed controls are also saved in a database to be used later in the *Maintenance Portal* UI or for generating reports.

4.2 Monitor Components

Complex Event Processing (CEP). The CEP detects misbehaviors of the STS, indicating potential threats to trustworthiness. It handles atomic events reported by STS-specific sensors, which are needed for monitoring trustworthiness. In different contexts (e.g., time intervals), these incoming events are aggregated to perform an initial analysis. To this end, a pre-defined configuration involves rules to detect patterns of related events. Based on the incoming sensor events and STS-specific detection rules, alerts are issued to the *Management* components for further threat analysis.

Trustworthiness Metric Tool. The Trustworthiness Metric Tool component serves as a repository for managing metrics to measure the trustworthiness of STS constituents, and estimate the user perception of trustworthiness. It allows browsing system quality attributes contributing to trustworthiness, as well as defining metrics details such as computation formulas. The tool also supports computing metrics at runtime, with reported trustworthiness properties as inputs. Metric values can be retrieved by other components, e.g., by the UI to generate reports covering longer time periods.

Trust Metric Estimator (TME). The TME is a Bayesian computational model that aims at estimating a user's trust level over time for a number of metrics defined by the *Trustworthiness Metric Tool*, e.g., trust with respect to reliability. These trust levels are calculated based on the personality of each trustor (retrieved from respective attributes, such as competency level and trust stance, stored in the *Customer Profile DB*), and system trustworthiness properties. The TME receives atomic trustworthiness-related events from the STS. In particular, a successful transaction increases trust and vice versa, while the magnitude of trust change depends on the user's segment (see [27] for more details on the trust computation, and the four segments that were found to have statistically significant differences). Fig. 3 shows the TME design.

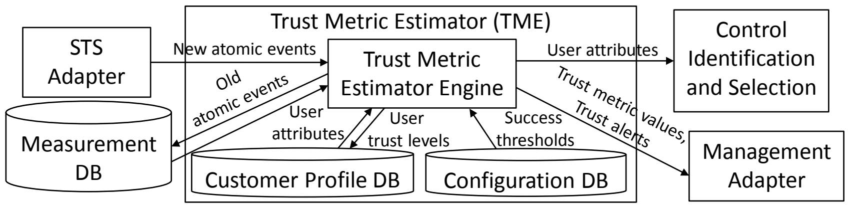


Fig. 3. Conceptual design of the Trust Metric Estimator

A particular transaction performed by a user is characterized as successful, or not, by comparing the atomic event value with the threshold value defined by the administrator for that metric, and eventually is stored in the *Measurement DB* (cf. Fig. 2). To this end, versatile trustworthiness characteristics (such as response time etc.) can be chosen. If the current trust level of any user in the system exceeds the thresholds set by the administrator in the *Configuration DB*, a trust alert is generated by the *TME Engine* and forwarded to the *Management*. The evolution of trust levels is also stored in the *Customer Profile DB*, and provided to the *Maintenance Portal* and the OCS.

User Behaviour Monitor (UBM). The purpose of the UBM is to continuously monitor and measure trust-related behaviors of individual users through respective sensors indicating these behaviors. It relates the behaviors to trust disposition of the user and to a model of trust in the system. Although the UBM supports any kind of STS, the sensors are system-specific and need to be configured accordingly. Each sensor needs to be configured with the following three parameters:

- A *low threshold* allows raising alerts in case the trust level estimated by a sensor drops under this limit, indicating a potential low trust perceived by the user.
- A *high threshold* is used for raising alerts in case the trust level exceeds the defined value, suggesting that the system is over-performing.
- Additionally, a *weight* is set, which is used to evaluate the overall trust level.

The UBM collects trust-related atomic events from different sources, stores these events in a persistent database, and performs an initial processing to aggregate them and compute metric values characterizing trust in terms of user behavior. When a certain trust measurement reaches a predefined threshold, which can be either too low or too high trust, a respective alert is generated and forwarded to the *Management*, which will then select an appropriate mitigation control to restore trust in the use of the system. The UBM consists of three main modules, as shown in Fig. 4:

- The *Generic (User) Trust Model* comprises a list of application-specific sensors, which are configured with the corresponding parameters mentioned above.
- The *Trust Estimator* processes input from the sensors, analyses the data, and consecutively issues alerts in the case of a trust violation.
- In the *User Trust Database* the trust history for every user is stored.

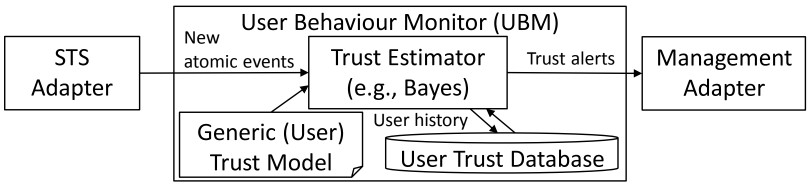


Fig. 4. Conceptual design of the User Behaviour Monitor

4.3 Management Components

Trustworthiness Evaluation (TWE). The TWE is responsible for identifying and classifying threats related to trustworthiness based on calculating threat likelihoods, as well as for determining appropriate control objectives. Threat and control identification is based on machine reasoning using a generic threat ontology that incorporates relevant knowledge, and an internal runtime model of all the different STS constituents and their behaviors. This model is incrementally updated at runtime. Background on the models used for trustworthiness evaluation can be found e.g. in [11]. Fig. 5 shows a simplified conceptual design of the TWE, comprising four main modules:

- A *Controller* handles incoming requests and maps them to the responsible module.
- The *Incremental Model Generator* incrementally updates the *Runtime STS Model* according to events reported by the *Monitor* (i.e., the CEP). Runtime event handling mechanisms are used to also reflect system topology updates (e.g., considering the deployment of controls) and changes of control objectives in real time.
- The *Vulnerability Finder* enforces control rules as defined in the *Threat Ontology* to classify trustworthiness threats into blocked or mitigated threats, secondary effects, or vulnerabilities, according to the presence of controls.
- The *Bayesian Estimator* analyses the likelihood of all threat activity given the reported trustworthiness behaviors of the STS. This quantitative threat analysis is based on a well-defined statistical model using Bayesian networks.

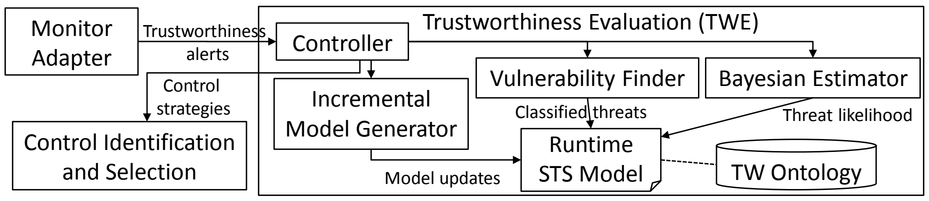


Fig. 5. Conceptual design of the Trustworthiness Evaluation

Trust Evaluation (TE). The TE receives alerts from the *Monitor* components TME and UBM, and analyses them by means of an internal, system-specific trust model to detect the current threats that may arise due to changes in user trust. Based on the threat evaluation, the TE proposes control strategies for the successive mitigation actions. To this end, an interface is used by the *Control Identification and Selection* to find active trust threats (querying the current runtime models) and propose a list of control strategies. The TE is composed of the following main modules (see Fig. 6):

- The *Trust Evaluation Controller* preprocesses the trust alerts from the *Monitor* to determine their relative types and the priority to handle them.
- The *Trust Event Finder* discovers trust-related threats based on the trust alerts. To this end, an application-specific representation of each user's expected behavior, i.e., the *User Behaviour State Machine Model*, is utilized. A misbehavior may indicate the presence of a threat due to a lack of user trust.
- The *Trust Reasoner* determines how to handle the threats discovered by the *Event Finder*. It proposes a list of applicable control strategies for subsequent mitigation.
- The *Trust Ontology* database keeps track of the application-specific trust terms that are used in the *User Behaviour State Machine Model*.

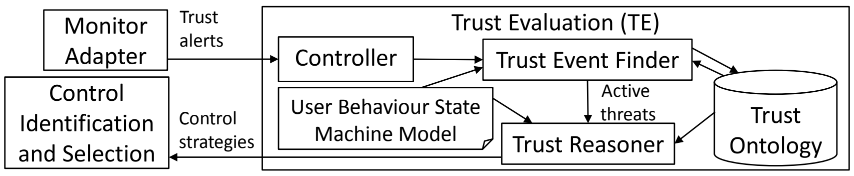


Fig. 6. Conceptual design of the Trust Evaluation

Optimal Control Selector (OCS). The OCS (a subcomponent of the *Control Identification and Selection*, see Fig. 7) suggests the most cost-effective control(s) to be deployed in order to deal with a threat regarding a trustworthiness misbehavior, and/or user trust concern. More specifically, it maximizes the probability that the metric, for which an alert was triggered, will have an acceptable value after a certain number of transactions, while keeping the expected costs low. A provider can maximize its expected profits using the approach below (see [28] for more details):

1. Estimate the initial trust level of all users in the service and for the particular metric associated to the incoming alert from the TE. These alerts are retrieved from the TME via the *Monitor Adapter*. This has to be performed for each user segment, rather than for each individual user.
2. Compute the minimum of successes necessary for each user segment to reach the initial trust level after a number of transactions (i.e., the optimization window).
3. The *Contingency Plan Engine* creates the so-called contingency plan for reaching the initial trust level in a cost-effective way by solving a dynamic programming problem and identifies the current optimal control. In order to do so, we need details on candidate controls (i.e., price and trustworthiness properties) from a marketplace providing alternative services that can be deployed as possible controls.

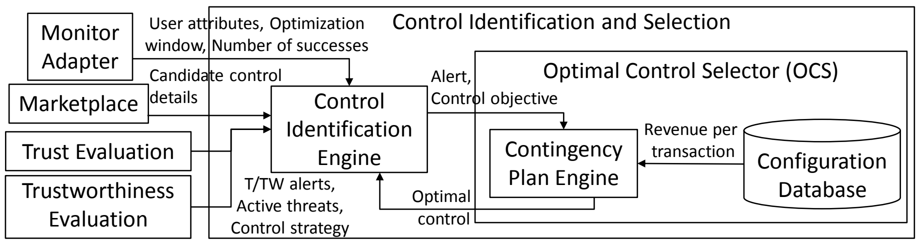

Control Identification and Selection
Monitor Adapter
User attributes, Optimization window, Number of successes
Marketplace
Candidate control details
Trust Evaluation
Trustworthiness Evaluation
T/TW alerts, Active threats, Control strategy
Control Identification Engine
Alert, Control objective
Optimal control
Optimal Control Selector (OCS)
Contingency Plan Engine
Revenue per transaction
Configuration Database

Fig. 7. Conceptual design of the Optimal Control Identification and Selection

4.4 Design of the Maintenance Portal User Interface

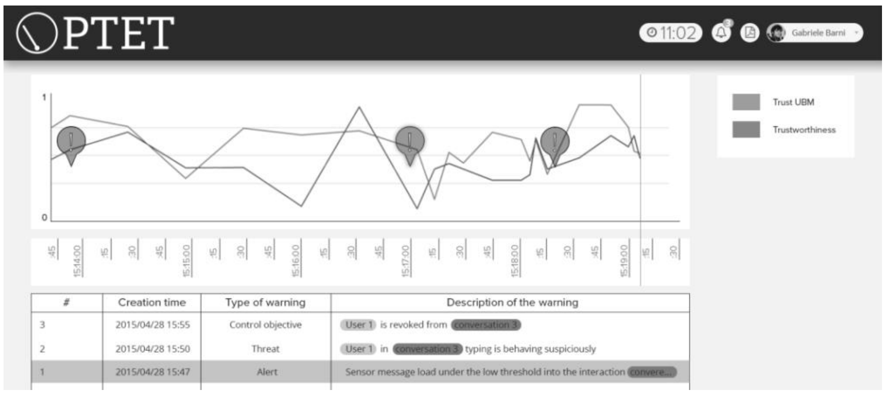
The *Maintenance Portal* shall provide the administrator information on the current and past state of both trust and trustworthiness, enabling her to detect any trust and trustworthiness violation, understand the root cause of the violation, and, finally, to approve and/or choose the most appropriate mitigation. In order to address these needs, the UI has been designed and implemented with the following capabilities:

1. Display all the necessary graphs to reproduce current and past trends of all the different trust and trustworthiness parameters relevant for runtime maintenance.
2. Provide the user with runtime trust and trustworthiness information on different levels of abstraction, starting from a general view of the overall trust and trustworthiness behaviors to more detailed graphs illustrating, e.g., trust of different users, or lower-level trustworthiness properties.
3. Notify the administrators anytime, no matter which page is actually displayed, of any relevant event about the status of the system (e.g., alerts and threats).
4. Visualize detailed information about these raised alerts and the detected threats, following any reported trust and trustworthiness violation.
5. Propose a list of applicable controls in order to mitigate the detected threats.
6. Allow the administrator to select and apply one of the proposed mitigation actions.

Fig. 8 shows the UI landing page. It shows the overall system trust and trustworthiness levels. This screen also shows the alerts to be handled. The administrator can browse more detailed levels, e.g., depicting the trustworthiness per quality attribute or a specific constituent of the STS. To avoid frequent pop-ups, the user is notified of new alerts using an icon on the top right (bell icon); only when the user clicks it, the complete alert information will be displayed. The notification table enables the user to immediately take action[1] by pressing the ‘Take Action’ button in case the situation requires mitigation. More details on each of the UI screens can be found in [12].

In addition to the feedback received via the UI, runtime reports (i.e., XML documents) can also be generated to provide offline feedback to system administrators on the STS’ trust and trustworthiness status during different time intervals, or to be consumed by other tools, such as an online software marketplace.

[1] Note that, based on configuration, the tool may select mitigation actions automatically, or query the administrator.



#	Creation time	Type of warning	Description of the warning
3	2015/04/28 15:55	Control objective	User 1 is revoked from conversation 3
2	2015/04/28 15:50	Threat	User 1 in conversation 3 typing is behaving suspiciously
1	2015/04/28 15:47	Alert	Sensor message load under the low threshold into the interaction convere...

Fig. 8. Main screen of the Maintenance Portal user interface

5 Application Example

The application example for demonstrating the trust and trustworthiness maintenance tool is based on a Secure Web Chat (SWC) system. The SWC addresses the need for trustworthy online communication, which may be vital in case of a cyber-attack, so that users of critical STS can ask for advice, and administrators can discuss appropriate actions or consult external experts. SWC users can create or join secure chat rooms to discuss critical topics and exchange files. Hence, the SWC is a complex STS consisting of software and hardware infrastructure, but also a multitude of human users. The SWC mainly faces the following trust and trustworthiness concerns:

- A high level of user trust: Although the usage of the SWC may be mandatory in case there is no other means for secure communication, a user with low trust in the SWC may not be able to contribute adequately to solving a cyber crisis.
- Accurate real-time communication: The chat room participants need to communicate efficiently in order to manage a cyber crisis. Furthermore, low performance or failures of the system will cause users mistrusting the SWC.

As mentioned, our initial evaluation aims at demonstrating and verifying our prototype tool and thereby showing the feasibility of our approach, as well as allowing further validation. We designed three evaluation scenarios to systematically focus on specific aspects of trust and/or trustworthiness maintenance, and the corresponding features of our prototype. Based on the SWC, we specified simplified input data (i.e., events reported from SWC-specific sensors), and determined the expected output of the tool components. We simulated the event stream to exercise specific, functionally related components and thereby invoke the involved functionalities separately.

In [29] we already briefly sketched our evaluation plan involving the three scenarios, which will be described in the following. In particular, we will explain the behavior, the use, as well as the responses and outcomes of the different tool components for illustration purposes. The results of the exemplary application show that the tool performs as expected, and sustain our confident that the tool will be useful in practice.

Trust Scenario. This scenario focuses on user trust in terms of trust-related behavior (see Section 3). Hence, it specifically evaluates the UBM and the TE. To monitor the evolution of user trust over time, the users have been segmented, and initial trust levels have been computed based on our trust computation approach sketched in [27].

Regarding the user behavior, we configured suitable UBM sensors reporting respective events. Based on the SWC use case, we made some assumptions for detecting trust-related user behaviors, in order to simplify the complex matter. In practice, trust monitoring will demand for more elaborate concepts (cf. [5]). Our approach allows for defining system-specific sensors and thereby tailoring or refining the user behavior monitoring to a specific system to be monitored. In our example, unusual user behaviors focus on the message activity of each user. Abnormal user behaviors indicate a lack of trust in the SWC and the other chat participants. Hence, the evaluation scenario involves the following two applications of trust maintenance:

- A SWC user in a chatroom raises many questions, and sends many messages. The input events reported by SWC-specific sensors carry the numbers of messages in general and questions in particular, which occurred in a given time period for each user. These events are evaluated against configured thresholds. For instance, in case a user raises more than five questions in a reporting period of two minutes, the UBM calculates a decrease in the current trust level. In contrast, e.g. less than three questions will reflect in an increase in the trust level.
- A SWC user participates very slowly, and mainly contributes very short messages. Based on respective system-specific monitoring data and thresholds, the UBM computes an increase or decrease in trust for all the monitored users in a chatroom. For instance, messages shorter than 30 characters will cause a decrease in trust.

The UBM continuously updates each user's trust level. Based on thresholds defined for the trustor segments (cf. [27]), it issues alerts to the TE, which updates the internal model, identifies low trust threats, and proposes suitable controls. In both situations described above, the following mitigations are proposed to the SWC administrator:

- Automatically notify the user having low trust of either unclearness regarding the user's motives, or about a check of the situation.
- Open a special communication channel (e.g., a separate chat room) allowing the administrator to discuss and solve the user's trust misbehavior.
- Exclude the abnormally behaving chat user from the conversation.

Trustworthiness Scenario. For service providers, it is essential to ensure the QoS at a level satisfactory for the service consumers, such as the SWC users. As mentioned, the SWC is faced with reliability and robustness concerns, as well as high availability. This scenario shows the following trustworthiness maintenance applications:

- Reliability and robustness are measured based on the SWC's ability to handle exceptions. Any occurring exception is reported by an atomic event that also indicates whether the exception was successfully handled. The CEP aggregates these events to calculate the ratio of recovered exceptions and the total number of exceptions. If this ratio is too low, the CEP reports an alert to the TWE. The TWE continuously updates its internal runtime model to reason about current threats, and asserts a software malfunction threat. Finally, the Mitigation interacts with the TWE to determine software patching as the control objective to mitigate the threat.

- To measure the availability of the SWC, the CEP aggregates "alive" notifications reported from suitable SWC sensors using pings. A simple detection rule is used to compare the current mean availability against a predefined threshold (e.g., 95%), and to alert the TWE about underperforming SWC services. To counteract the related threat of an under-provisioned service, scalability is the suggested control objective to counteract the threat, e.g., by load balancing or adding resources.

Optimal Control Selection Scenario. This scenario demonstrates the identification of the most cost-effective control to mitigate a threat, in case there is a set of suitable controls available. It focuses on the second aspect of our trust maintenance approach (see Section 3), i.e., it deals with monitoring user trust based on the perceived level of trustworthiness. Hence, threats pertaining to user trust are mapped to control strategies affecting the trustworthiness of the system by restoring the system's QoS.

In this scenario, we consider trustworthiness (and the effect of its perception on user trust) in terms of the response times of the SWC for processing and delivering chat messages. Similar to the trust scenario, in a first monitoring step the trust levels of the users (which are grouped into four trustor segments) is continuously updated. Thresholds are defined in the TME, so that it can issue low trust alerts to the TE.

The TE then identifies a control strategy, i.e., a set of potential controls to be considered for mitigating the threat. In this application example, we defined three different options for restoring trustworthiness by substituting underperforming services. For each service that can serve as a substitute, its trustworthiness (in terms of response time), and the associated cost are defined. A trusted software marketplace provides the relevant metric values for each of these controls so that they can be compared. The possible controls are passed to the OCS component, which then identifies the optimal control. The suggested control may change over time, when the trust-decreasing effect of high response times is active over a longer period.

6 Conclusion and Future Work

In this paper, we described a unified approach complementing runtime trust and trustworthiness maintenance, and a corresponding tool prototype. Our unified approach specifically addresses the challenge of relating and coordinating objective system trustworthiness and subjective user trust at runtime by presenting system administrators comprehensive information about both, and thereby supporting the decision-making process for maintaining complex STS. Our trustworthiness maintenance is based on observing measurable system properties that contribute to the trustworthiness of the STS, while the trust maintenance relies on quantifying the subjective user trust through monitoring user behavior and estimating the perception of system trustworthiness characteristics. The tool prototype demonstrates the technical feasibility, and allows further investigating the validity of our approach. An initial evaluation illustrates a potential application of the tool, and shows that the different prototype components work as expected in different scenarios involving different functionality.

Future work should focus on a more elaborate validation of our approach. To this end, the tool prototype could be applied to a real industry case example in order to further evaluate the benefits of our approach and discuss it with potential stakeholders. This will contrast using our approach with using existing tools. Furthermore, general interdependencies between trust and trustworthiness can be examined using the tool. The resulting information may be used to discover potential for extensions or refinements of our approach, and, ultimately, to define concepts and techniques for balancing trust and trustworthiness at runtime.

Acknowledgements. This work was supported and funded by the EU FP7 project OPTET (grant no. 317631).

References

1. Miller, K.W., Voas, J.: The Metaphysics of Software Trust. IT Pro 11 (2), 52–55 (2009)
2. Hasselbring, W., Reussner, R.: Toward Trustworthy Software Systems. In: IEEE Computer 39 (4), 91–92 (2006)
3. Gol Mohammadi, N., Paulus, S., Bishr, M., Metzger, A., Könnecke, H., Hartenstein, S., Weyer, T., Pohl, K.: Trustworthiness Attributes and Metrics for Engineering Trusted Internet-Based Software Systems. In: Helfert, M., Desprez, F., Ferguson, D., Leymann, F. (eds.) CLOSER 2013. CCIS, vol. 453, pp 19–35. Springer, Heidelberg (2014)
4. Bandyszak, T., Gol Mohammadi, N., Bishr, M., Goldsteen, A., Moffie, M., Nasser, B.I., Hartenstein, S., Meichanetzoglou, S.: Cyber-Physical Systems Design for Runtime Trustworthiness Maintenance Supported by Tools. In: REFSQ Workshops, pp. 148–155. CEUR (2015)
5. Muir, B.M.: Trust in Automation: Part I. Theoretical Issues in the Study of Trust and Human Intervention in Automated Systems. In: Ergonomics 37 (11), 1905–1922 (1994)
6. Clark, K.P., Warnier, M.E., Quillinan, T.B., Brazier, F.M.T.: Secure Monitoring of Service Level Agreements. In: 5th Int. Conf. on Availability, Reliability, and Security (ARES), pp. 454–461. IEEE (2010)
7. Zhao, S., Wu, G., Li, Y., Yu, K.: A Framework for Trustworthy Web Service Management. In: 2nd Int. Symp. on Electronic Commerce and Security, pp. 479–482. IEEE (2009)
8. Shaikh, J., Fiedler, M., Collange, D.: Quality of Experience from User and Network Perspectives. In: Annals of Telecommunications 65 (1), 47–57 (2010)
9. Bouch, A; Bhatti, N; Kuchinsky, A.J.: Quality is in the Eye of the Beholder: Meeting Users' Requirements for Internet Quality of Service. In: Proc. CHI'00, pp. 297–304. ACM (2000)
10. Gol Mohammadi, N., Bandyszak, T., Moffie, M., Chen, X., Weyer, T., Kalogiros, C., Nasser, B., Surridge, M.: Maintaining Trustworthiness of Socio-Technical Systems at Run-Time. In: Eckert, C., Katsikas, S.K., Pernul, G. (eds.) TrustBus 2014. LNCS, vol. 8647, pp. 1–12. Springer, Heidelberg (2014)
11. Goldsteen, A., Moffie, M., Bandyszak, T., Gol Mohammadi, N., Chen, X., Meichanetzoglou, S., Ioannidis, S., Chatzidiam, P.: A Tool for Monitoring and Maintaining System Trustworthiness at Runtime. In: REFSQ Workshops, pp. 142–147. CEUR (2015)

12. OPTET Consortium: D6.4.2 – Measurement and Management Tools (2nd release). Technical report, http://www.optet.eu/wp-content/uploads/deliverables/OPTET_WP6_D6.4.2_Measurement_and_Management_tools_2nd_Release_V2.0.pdf
13. Alhamazani, K., Ranjan, R., Mitra, K., Rabhi, F., Prakash Jayaraman, P., Khan, S.U., Guabtni, A., Bhatnagar, V.: An Overview of the Commercial Cloud Monitoring Tools: Research Dimensions, Design issues, and State-of-the-Art. Computing 97 (4), 357–377 (2015)
14. Elshaafi, H., McGibney, J., Botvich, D.: Trustworthiness Monitoring and Prediction of Composite Services. In: 2012 IEEE Symp. on Computers and Communications, pp. 000580–000587. IEEE (2012)
15. Lenzini, G., Tokmakoff, A., Muskens, J.: Managing Trustworthiness in Component-Based Embedded Systems. Electronic Notes in Theoretical Computer Science 179, 143–155 (2007)
16. Yu, T., Zhang, Y., Lin, K.: Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints. ACM Transactions on the Web 1(1), 1–26 (2007)
17. Leichtenstern, K., Bee, N., Andre, E., Berkmuller, U., Wagner, J.: Physiological Measurement of Trust-Related Behavior in Trust-Neutral and Trust-Critical Situations. In: Wakeman, I., Gudes, E., Damsgaard Jensen, C., Crampton, J. (eds.) IFIPTM 2011. IFIP AICT, vol. 358, pp. 165–172. Springer, Heidelberg (2011)
18. BalaBit IT Security Blindspotter, https://www.balabit.com/network-security/blindspotter
19. ITU: Vocabulary and Effects of Transmission Parameters on Customer Opinion of Transmission Quality, Amendment 2, ITU-T Rec. P.10/G.100. ITU (2006)
20. Huang, T.-Y., Chen, K.-T., Huang, P., Lei, C.-L.: A Generalizable Methodology for Quantifying User Satisfaction. In: IEICE Trans. Comm. E91-B (no.5), 1260–1268 (2008)
21. Vakili, A., Grégoire J.-C.: QoE Management for Video Conferencing Applications. Computer Networks 57, 1726–1738 (2013)
22. Saladino, D., Paganelli, A., Casoni, M.: A Tool for Multimedia Quality Assessment in NS3: QoE Monitor. Simulation Modelling Practice and Theory 32, 30–41 (2013)
23. Klaue, J., Rathke, B., Wolisz, A.: EvalVid – A Framework for Video Transmission and Quality Evaluation. In: Kemper, P., Sanders, W.H. (eds.) TOOLS 2003, LNCS, vol. 2794, pp. 255–272. Springer, Heidelberg (2003)
24. Fiedler, M, Hossfeld, T., Tran-Gia, P.: A Generic Quantitative Relationship between Quality of Experience and Quality of Service. IEEE Network 24 (2), 36–41 (2010)
25. Zhang, J., Ansari, N.: On Assuring End-to-End QoE in Next Generation Networks: Challenges and a Possible Solution. IEEE Communications Magazine 49 (7), 185–192 (2011)
26. Surridge, M., Nasser, B., Chen, X., Chakravarthy, A., Melas, P.: Run-Time Risk Management in Adaptive ICT Systems. In: Proc. of ARES 2013, pp. 102–110. IEEE (2013)
27. Kanakakis, M., van der Graaf, S., Kalogiros, C., Vanobberghen, W.: Computing Trust Levels Based on User's Personality and Observed System Trustworthiness. In: Conti, M., Schunter, M., Askoxylakis, I. (eds.) TRUST 2015. LNCS, vol. 9229, pp. 71–87. Springer, Heidelberg (2015)
28. Kalogiros, C., Kanakakis, M., van der Graaf, S., Vanobberghen, W.: Profit-Maximizing Trustworthiness Level of Composite Systems. In: Tryfonas, T., Askoxylakis, I. (eds.) HAS 2015, LNCS, vol. 9190, pp. 357–368. Springer, Heidelberg (2015)
29. Bishr, M., Heinz, C., Bandyszak, T., Moffie, M., Goldsteen, A., Chen, W., Weyer, T., Ioannidis, S., Kalogiros, C.: Trust and Trustworthiness Maintenance - From Architecture to Evaluation. In: Conti, M., Schunter, M., Askoxylakis, I. (eds.) TRUST 2015, LNCS 9229, pp. 319–320, Springer, Heidelberg (2015)