A Framework for Evaluating the End-to-End Trustworthiness

Nazila Gol Mohammadi, Torsten Bandyszak, Thorsten Weyer
paluno - The Ruhr Institute for Software Technology,
University of Duisburg-Essen, Essen, Germany
{nazila.golmohammadi, torsten.bandyszak,
thorsten.weyer}@paluno.uni-due.de

Costas Kalogiros, Michalis Kanakakis
Athens University of Economics and Business,
Athens, Greece
{ckalog, kanakakis}@aueb.gr

***Abstract*—Trustworthiness of software and services is a key concern for their use and adoption by organizations and end-users. Trustworthiness evaluation is an important task to support making informed decisions for both providers and consumers, i.e., for selecting components from a software marketplace. An analysis of the state of the art in software evaluation technologies motivated us to develop an evidence-based approach for trustworthiness evaluation. Most of the literature evaluates trustworthiness by focusing on a single dimension (e.g., from the security perspective) while there are limited contributions towards multifaceted and end-to-end trustworthiness evaluation. Our analysis reveals that there is a lack of a comprehensive framework for comparative, multi-faceted end-to-end trustworthiness evaluation, which takes into account different layers of abstractions of both the system topology and its trustworthiness. In this paper, we provide a framework for end-to-end trustworthiness evaluation using computational approaches, which is based on aggregating certified trustworthiness values for individual components. The resulted output supports in the definition of trustworthiness requirements for a software component to be locally developed and eventually integrated within a system, as well as, trustworthiness evidences for a composite system before the actual deployment. Thereby supports the designer in analyzing the end-to-end trustworthiness values. An application example illustrates the application of the framework.**

Keywords— Socio-Technical-System; Computational Evaluation; Trustworthiness; Metrics; End-to-End Evaluation

I. INTRODUCTION

Modern information and communication technologies enabled significant improvements and facilitated the growth of Socio-Technical systems (STS) and their integration in our daily life. STS comprise information systems, software and computer systems, mechanical parts, as well as organizations and humans that use the system in order to achieve a goal [1] [2]. An STS is shaped by technologies and services that contain software as core elements. The users of these STS depend on these software services for performing their activities in business or organizational settings, and in their social life. Therefore, trustworthiness of these systems is a key factor for user to trust and adopt them. The software elements in these systems should be designed and manufactured in such a way that they sophisticatedly satisfy trustworthiness requirements. It is not only essential to use constructive quality assurance techniques, such as best practices for development processes [3] [4], but also to analytically evaluate the trustworthiness of a desired system early in the design phase. Trustworthiness can be seen as an objective system property reflecting its ability to assure that it will perform as expected [5], e.g., a elderly user of a health care service on the web, needs confidence that it will meet her usability expectations, whereas organizations require confidence about their business critical data. In order to achieve objectivity, we need to measure certain system qualities that are relevant to achieve trustworthiness. To this end, metrics can be used in order to quantify trustworthiness attributes [6]. Furthermore, measurements and corresponding metric values can be used as evidences for certifying a certain quality level.

Systems are often composed of existing software services or components that are certified and provided on a software market place (cf., e.g., [7]). Component-based development [8] poses the challenge of considering different component structures for determining the "End-to-End" (E2E) trustworthiness of the overall system. Different certified metric values of all the involved components have to be aggregated considering the specific system topology, in which they are embedded. Particularly, redundancy is often introduced in system design, for instance, as a means to increase trustworthiness in terms of higher reliability or availability.

Another challenge consists in aggregating the resulting E2E trustworthiness values on different levels of granularity or abstraction, e.g., on the level of trustworthiness attributes or even for overall trustworthiness as a very high-level trustworthiness indication. Despite a large number of indications in the literature in evaluation and documentation of the design decisions based on these evaluation results, the E2E evaluation of multi-faceted trustworthiness remains an open research. There are approaches that merely focus on e.g., reliability [9]. However, trustworthiness is rather a broad-spectrum term with notions including reliability, security, performance, and usability as parts of trustworthiness attributes [6]. Therefore, a holistic taxonomy of software quality attributes that contribute to trustworthiness and corresponding metrics (presented in our previous work [6]) is used as basis.

Thus, there are two dimensions that need to be taken into account when evaluating the overall system trustworthiness; the first dimension is the overall system structure while the second is the level of granularity of E2E calculation, e.g., regarding a hierarchy of trustworthiness attributes, and sets of different metrics. Our approach builds upon available formulas

that consider different system structures for calculating overall trustworthiness.

This paper addresses the problem of evaluating the overall trustworthiness of online STS, with a particular focus on software assets that are accessible via an online marketplace. Some software Marketplace allows integrators and service providers to deploy a new composite system by selecting system assets and compose them in order to create a new system based on their trustworthiness certificates [7] [10]. We use different metrics to quantify system trustworthiness attributes, and use the trustworthiness metric values in the certificate of each software component as parameters for calculating the overall E2E trustworthiness. To this end, workflow models serve as adequate abstractions to specify sequences of assets that are involved in achieving some task. Based on these low-level E2E trustworthiness values, more aggregate values can be calculated, eventually resulting in an overall trustworthiness value. E2E values can be used as evidence of the system's trustworthiness, and to compare different candidate system compositions.

We propose a framework that supports designers in composing E2E formulas and performing the trustworthiness evaluation process. This framework includes metric skeletons and templates, as well as guidance for determining aggregated values on different abstraction layers. As an initial evaluation, we present the application of our approach to evaluate the E2E trustworthiness of an exemplary system from the Ambient Assisted Living (AAL) domain. We also show how the proposed framework supports E2E trustworthiness evaluation.

The remainder of the paper is structured as follows: In Section II, we present the fundamentals and definitions of the main concepts. A brief overview of existing techniques for evaluating trustworthiness of software is provided in SectionIII. Section IV describes our approach in evaluating E2E trustworthiness. Section I presents an application example using two scenarios. Section VI concludes the paper and elaborates on future work.

II. Fundamentals

This section presents the fundamental concepts that form the basis for our approach.

A. Trustworthiness Attributes and Metrics

We analysed software quality attributes and their contribution to trustworthiness, and presented a comprehensive set of trustworthiness attributes that should be considered in the design of trustworthy STS [6]. This approach covers a wide range of quality attributes instead of only focusing on e.g. security. The concrete trustworthiness attributes are domain and application dependent, e.g., in health care applications, the set of attributes which have primarily been considered consists of availability, confidentiality, integrity, maintainability, reliability and safety, but also performance and timeliness. Trustworthiness attributes can be classified and aggregated by higher-level trustworthiness categories, such as "dependability", which may be concretized by certain attributes such as "availability". In order to quantify trustworthiness attributes, metrics can be systematically derived. Hence, trustworthiness metrics serve the purpose of objectively identifying and measuring real-world properties that characterize and contribute to trustworthiness attributes. Regarding software-intensive systems, Kan in [11] distinguishes three types of software metrics: product metrics, process metrics, and project metrics. In this paper, we focus on product metrics focusing on characteristics of software products. According to IEEE 1061 [12], a software quality metric is a "function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality". Thus, metrics allow for measuring and quantifying certain trustworthiness attributes by means of more concrete properties of a system.

B. Component-based System Design

An STS consists of several assets, i.e., anything of value in an STS [13], including physical, technical or logical parts, as well as humans. An asset is an abstract, basic building block of a system that may manifest in different implementations (i.e., asset instances from different vendors).

Component-Based Software Engineering aims at extensively reusing existing components in software development, and focuses on e.g. components and interface models [1]. In the area of Service-Oriented Architectures (SOA), software components, which are independent of their environment, and loosely coupled, are used in order to build systems that support business processes. The term "Workflow" is used in the context of Web Service composition. Workflows describe business processes, and can be related to software services that support them [1]. BPMN, a modelling language for representing business processes and control flows, can be used for web service orchestration [14]. Concerning the modelling of component-based systems, the Reliability Block Diagram, as used in Reliability Engineering for complex systems, allows the designer to model different composition types, i.e., series, parallel (for modelling redundancy), and combined series-parallel structures [9]. Related to web service composition, there are some more specialised modelling and description languages such as BPEL [15].

Software Marketplaces, such as the Amazon Web Services Marketplace [16], provide platforms for distributing and offering software services to organisations. In order to address the problem of trustworthiness of the offered services, certification is a mechanism to guarantee certain levels of service [17]. For instance, Ali et al. present a Marketplace system that enables the provision of security certificates [1]. The concept of a Trustworthy Sofware Marketplace [7] incorporates machine-readable security certificates, and allows for matching these trustworthiness evidences to user requirements. We argue that the certification approach can be extended to include multi-faceted trustworthiness attributes that are quantifiable by means of metrics [5].

III. Related Work

Service composition and evaluation with respect to trustworthiness or quality of service, has been researched. Klatt et al. propose to use a service quality prediction of composed services in order to support service composition considering service quality [18]. Quality evaluation is also an integral part of the service composition framework proposed by Liu et al. in [19]. Elshaafi, et al. present an approach towards measuring the trustworthiness of a service composition with focus on run-time monitoring [20]. They provide formulas that allow for

calculating the trustworthiness (in terms of reputation, reliability, and security) of composite services, taking into account several service composition constructs such as sequence, parallel, loop, choice, discriminator, and multichoice-multimerge patterns. Zhao et al. propose a framework for trustworthy web service management, which also involves formulas for aggregating the availability, reliability, and response time of services composed in sequence, parallel, conditional, and loop structures [21]. Other approaches, such as [22] focus on reputation by aggregating service ratings in order to determine a provider's rating. Quality of Service (QoS) aggregation can be applied in order to determine the QoS of a web service workflow based on the QoS of each involved or executed web service [23], [24]. Cardoso et al. utilize graph reduction mechanisms and respective formulas for aggregating time, cost, and reliability of service workflows [22]. Workflow composition patterns and aggregation schemes are also given in [24]. Hwang et al. propose a probabilistic approach for estimating the QoS of service compositions, which is based on more elaborate metrics, and addresses uncertainty given for QoS values [23]. They consider sequence, parallel, choice, discriminator, and loop structures in addition interleaved parallel, multiple choices, and m-out-of-n constructs. Related to the use of metrics, Wang and Crowcroft distinguish additive, multiplicative, and concave metrics for QoS routing, which can be considered as a problem that also applies to service composition [25]. Raheja and Gullo considered that the reliability of the whole system depends on the reliability of its components, thus, formulas that represent the different component structures are used in order to calculate the overall reliability [9].

Although the related approaches summarized above support a wide range of system structures, they focus on a limited set of trustworthiness metrics neglecting the system trustworthiness dimension previously described. Furthermore, we identified the need for establishing a comprehensive framework that supports a large set of trustworthiness metrics. More specifically, each trustworthiness metric is mapped to a metric type (multiplicative, concave, and additive) and has either a positive or a negative interpretation (whether higher values are desirable or not). For example, while both the availability and the error rate are of multiplicative type, the former has a positive interpretation while the latter has a negative one. Furthermore, each trustworthiness metric belongs to one trustworthiness attribute. The above information, together with the system structure, is used to calculate the overall trustworthiness metric. Even though we restrict to sequential topologies we can support more complex structures by allowing redundancy in specific asset instances (namely parallel and "k out of N" constructs).

IV. Framework for Evaluating End-to-End Trustworthiness

This section describes our E2E trustworthiness evaluation framework. We build upon existing approaches towards trustworthiness attribute classification, evaluation, and formulas for composite system structures, and unify them into a comprehensive framework that provides system designers with guidance for evaluating multi-faceted trustworthiness on different layers of abstraction. The aim of the framework is to facilitate the evaluation process, and to structure evaluation reports as the basis for selecting a certain design alternative.

Our framework covers two dimensions of E2E trustworthiness evaluation, as depicted in Fig. 1. On the one hand, the structure or topology of the entire software system involves many different assets that participate in a certain control or data flow relation to support a business process or to achieve some business goal. This structures is described in terms of workflows. For instance, parallel or redundant structures are often used to increase trustworthiness properties such as performance, reliability, or availability. These different aspects of system structures need to be taken into account when determining trustworthiness values of entire systems, and can be abstracted by focusing on a complete system which may be characterized by multiple workflows. On the other hand, the trustworthiness of both single software services and overall system structures can be evaluated on different levels of abstraction or granularity. For instance, at the lowest level, metrics are used to provide detailed evidences of specific trustworthiness properties, while these values have to be aggregated on the more abstract level of trustworthiness attributes such as "availability". The highest level of trustworthiness granularity provides an overall trustworthiness value for the whole system.

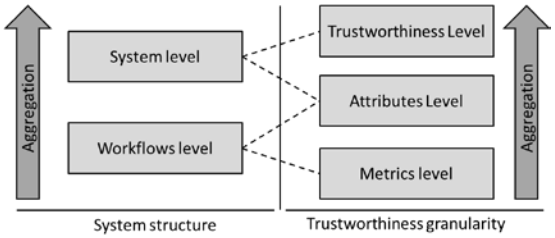

Fig. 1. Overview of the end-to-end trustworthiness evaluation framework.

The framework consists of two parts. First, an ontology that provides general concepts for E2E trustworthiness evaluation is presented. These concepts form the basis for establishing calculation-based trustworthiness evaluation of composite system structures on different levels of granularity. Second, we describe our approach for objectively evaluating E2E trustworthiness of a whole system's quality attributes, which aggregates the partial trustworthiness measurements of each asset instance composing a STS. Specifically, we describe abstraction mechanisms and a related process of successively aggregating trustworthiness values on different levels of granularity, which takes into account the system structure or topology. In the following, we describe the parts of the framework in more detail.

A. *Ontology for Design-Time End-to-End Trustworthiness Evaluation*

In order to establish a sound theoretical fundament for our E2E trustworthiness evaluation approach, we define some basic concepts that need to be considered when assessing the multi-faceted trustworthiness of composite systems. Fig. 2 shows the ontology of our E2E trustworthiness evaluation approach, which defines the relevant concepts as well as their relations.

The abstract concept of "components" of STS is represented by "Assets" and "Asset Instances". This distinction is necessary to differentiate between general and abstract

building blocks and concrete implementations that may participate in a redundancy relation.

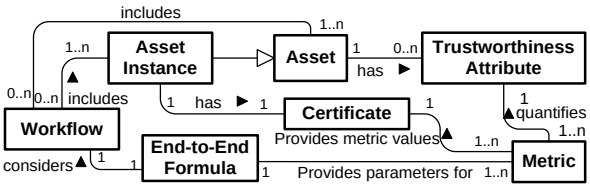

Fig. 2. Concepts of End-to-End Trustworthiness Evaluation.

An asset instance is a concrete manifestation of an asset (also denoted "asset category"). For example, "DBMS_1" could be a software service offered on the Marketplace as an instance of the asset "Database Management System". Certificates for asset instances are provided by a Certification Authority that evaluates a (software) implementation in order to confirm that it meets some trustworthiness goals. A certificate describes all observed trustworthiness properties of the software, as well as related evidence in terms of certified metric values (cf. [26]). A Workflow is a model that specifies the set of asset instances as well as cardinality and their interrelations, e.g., in the control flow of performing some business process.

The "Workflow" concept is an appropriate abstraction mechanism to focus on the aspects that are necessary for determining E2E trustworthiness formulas. More details on the information that is (graphically) modelled in a Workflow are presented in next subsection. An End-to-End Formula is a template or function that allows for calculating the trustworthiness of composite system structures represented in terms of Workflows. It requires metric values for each involved asset instance as arguments, and returns one value that characterizes the trustworthiness of the whole workflow.

In the following we describe our approach and the way that we use the described elements for evaluating trustworthiness of composite systems. Specifically, we elaborate on how an E2E formula as the central artifact is created, used in order to evaluate a system with aggregation of trustworthiness values at different levels of granularity.

B. End-to-End Trustworthiness Computation

This section presents our approach towards calculating E2E trustworthiness using the introduced framework. Specifically, we describe the steps of an evaluation process that takes into account different system structures as well as different levels of granularity. First, we show how adequate models are created in order to depict system and redundancy structures. Then, we describe how aggregation mechanisms are used in order to abstract from certain trustworthiness details in order to eventually derive an overall system trustworthiness value.

1) Workflow Modelling and End-to-End Formula Creation

The computational approach towards E2E trustworthiness evaluation relies upon the availability of metric values for each asset of the system as a means to quantitatively express trustworthiness. The metric values can be found, for instance, in certificates of the asset instances that are available on a marketplace, as described in Section II. Thus, the E2E computation is performed for concrete instances of the general assets that build up an abstract system.

Depending on the characteristics of its intended usage scenarios, a system can have arbitrary structures (sequential, tree, network, etc.), where the nodes could be seen as the building blocks of that system. A workflow is a specific composition (or sequence) of asset instances that are invoked and orchestrated in order to achieve a certain goal or to support some business process. A graphical workflow model (i.e., the workflow graph) aims at guiding the evaluation process by modelling and determining which objects (i.e., asset instances) are functionally connected and should thus be evaluated together in an E2E configuration. Hence, a system can be described by multiple workflows and respective graphs. Each workflow determines a particular part of the system that is on focus of evaluation, and contains vital redundancy information. An example of such a workflow graph is illustrated in next section. We propose to represent the following information in an appropriate workflow model:

- *(Sub-)System topology*: the topology includes the asset categories, and their relations.
- *Assets and asset instances*: Assets are abstract building blocks of a system, while asset instances denote concrete implementations or realizations of them.
- *Start and end node*: E2E trustworthiness evaluation requires the definition of two end nodes (i.e., system assets) as a starting and end node of a workflow sequence.
- *Redundancy group and type*: In addition to the interaction relations of asset instances, it is also necessary to model the redundancy among several asset instances of the same asset. A redundancy group contains a number of asset instances that participate in some kind of redundancy relation in order to e.g. increase the availability of the provided service. The redundancy type describes the minimum number of asset instances in a certain redundancy group that will be required to successfully process a request, e.g., any one of four ("OR" type), two out of four , or all four ("AND" type).

Three types of E2E metrics have been defined in the literature [25] [27]: the additive metrics (e.g., cost, the response time), the multiplicative metrics (e.g., mean availability) and the concave metrics (e.g., encryption key length). The metrics type has to be considered when determining the respective E2E formula. Table I provides skeletons of the mathematical formula that would be constructed for computing the trustworthiness value of a single asset (or asset category) j. Such an asset category is assumed to be consisting of $i = 1, \ldots, n$ asset instances, where m_i is the trustworthiness metric value that characterizes the trustworthiness of the i-th asset instance, and consequently appears in its trustworthiness certificate. Depending on the metric type (concave, multiplicative, or additive) as well as the metric target type, we get a different formula, e.g., concave metrics depend on the bottleneck asset instance and thus the minimum or maximum of the asset instance metric values is needed (e.g., the asset employing the smallest encryption key length). We should note that the formulae appearing for multiplicative metrics refer to the K-out-of-N case, which can be used to create the rest "extreme" constructs as well. More specifically, if $K = 1$ then it refers to the "OR" construct, while if $K = N$ we get "AND".

For simplification and better readability of the multiplicative formula skeleton, we assume that all asset instances belong to the same asset category, i.e., $m_i = m$ for

$i = 1, \ldots, n$, in contrast to the general case where $m_i <> m_j$. The rationale is that we consider all the combinations where at least K asset instances will complete a certain task. Thus, in the example of an asset category composed of three asset instances following the 2-out-of-3 construct we would consider four cases. Following a binary representation, where 0 refers to the event where a certain asset instance is not able to complete the task and, the following cases would be considered for metrics targeting at higher values: 011, 101, 110 and 111. Note that the formula skeleton for additive metrics is valid only for sequential compositions that have no redundancy.

Given that a workflow usually contains more than one asset category, the next step is to compose the E2E formula, denoted e. Table I provides a skeleton of the formula for all asset categories, say $j = 1, \ldots, k$ depending on the metric type.

TABLE I. COMPOSITION OF FORMULAS FOR CALCULATION OF ASSET CATEGORY TRUSTWORTHINESS AND END-TO-END TRUSTWORTHINESS.

Metric Type	**Metric Target Type**	**Asset Category Redundancy Type**	**Formula of Asset Category *j***	**Formula for E2E TW Metric**
Concave	Higher values	-	$a_j = \min_i m_i$	$e = \min_j a_j$
	Lower values		$a_j = \max_i m_i$	$e = \max_j a_j$
Multiplicative	Higher values	AND	$a_j = \prod_{i=1}^{n} m_i$	$e = \prod_j a_j$
		OR	$a_j = 1 - \prod_{i=1}^{n} (1 - m_i)$	
		K-out-of-N	$a_j = \sum_{i=k}^{n} \binom{n}{i} m^i * (1-m)^{n-i}$	
	Lower values	AND	$a_j = \prod_{i=1}^{n} m_i$	$e = 1 - \prod_j a_j$
		OR	$a_j = 1 - \prod_{i=1}^{n} (1 - m_i)$	
		K-out-of-N	$a_j = \sum_{i=k}^{n} \binom{n}{i} m^i * (1-m)^{n-i}$	
Additive	Both	-	$a_j = \sum_{i=k}^{n} m_i$	$e = \sum_j a_j$

The formula skeletons provide valuable guidance for representing different system structures and asset redundancy types in the form of a mathematical model for calculating E2E trustworthiness metric values with respect to related metrics. Metric values of single asset instances are then used as parameters for the E2E metrics that have been defined based on the workflows. In particular, an E2E metric value is derived for each workflow of the system, and each provided metric.

2) Aggregation of Trustworthiness Values

So far, the focus of evaluation was limited to a certain number of separate workflows, and individual metrics. As mentioned above, our E2E trustworthiness evaluation framework also considers different levels of granularity of the values describing the trustworthiness of a system. To this end, the concept of trustworthiness attributes is an appropriate means to abstract from different metrics that may be available for a certain attribute in the first place. Since the resulting metric values still pertain to certain workflows, they can be aggregated by focusing on a trustworthiness attribute related to the whole system, which may be characterized by multiple workflows. Calculating the minimum of all the different values pertaining to the workflows seems an adequate mechanism and in order to guarantee consistency the metrics where lower values are desirable are transformed into higher ones. Another approach could be determining the weighted average among the different values. Finally, the last step is to abstract from multiple workflows and calculate one overall E2E value. To this end, the designer can specify weights for each attribute into account and calculate one overall E2E value using a weighted average. Fig. 3 illustrates the steps in aggregating trustworthiness values on different levels of granularity.

To summarize, our approach allows for calculating E2E trustworthiness metric values on the following layers of abstraction:

- *E2E values per Workflow and Metric*: Given a workflow and a particular metric that can be used to estimate a certain trustworthiness attribute, we calculate an E2E metric using the E2E formula skeleton.
- *E2E values per Workflow and Attribute*: For determining the E2E value related to a certain workflow and trustworthiness attribute, the minimum value of all E2E values that are available for each of the metric pertaining to that attribute, is calculated.
- *E2E values per Attribute*: The E2E value per trustworthiness attribute is determined by calculating the minimum value for all the given workflows, related to this attribute.
- *An E2E value per system* (overall E2E trustworthiness): In order to calculate one E2E trustworthiness metric value for the whole system described by several trustworthiness attributes and workflows, weights are specified by the designer for each trustworthiness attribute.

V. APPLICATION EXAMPLE

This section provides an overview of a case example that demonstrates the application of our E2E trustworthiness evaluation framework. The example is taken from the domain of Ambient-Assisted Living (AAL). The exemplary Fall Management System monitors an elderly person at his or her home with respect to emergency situations, such as a fall. Detected emergency situations are reported to a central alarm handling service that will decide upon the actions that can be taken. Depending on the severity of the emergency, relatives can be notified, or ambulances requested. Fig. 4 shows an exemplary design-time model of the Fall Management System, and includes the main (software) components.

Using the E2E trustworthiness evaluation framework as guidance, a system designer or composer is supported in

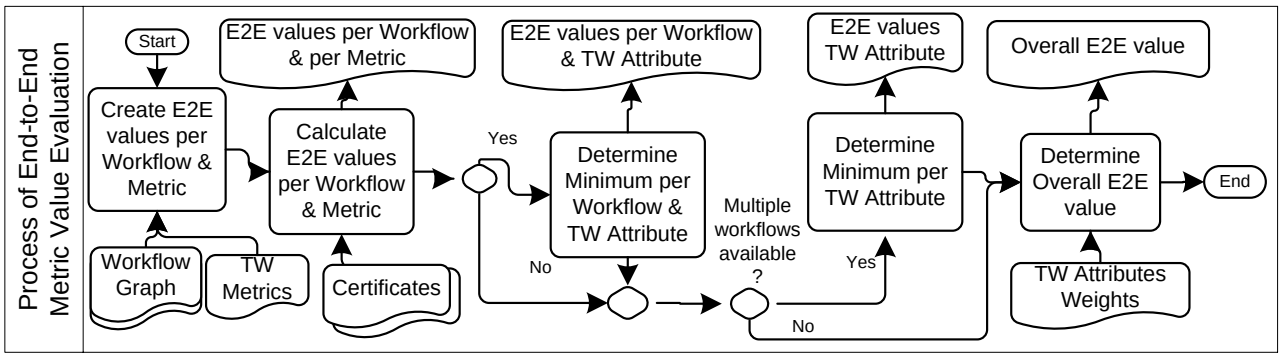

Fig. 3. *Process of Calculating End-to-End Metric values.*

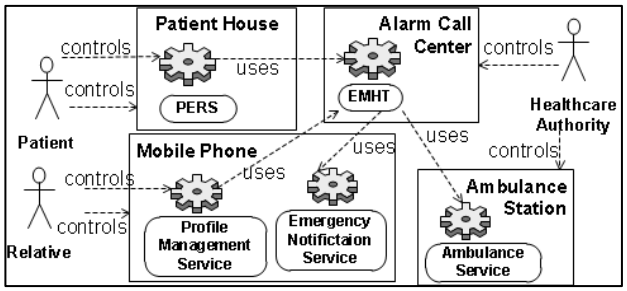

Fig. 4. Design-time system model of the Fall Management System

making informed decisions on the system configuration regarding trustworthiness requirements. The goal is to evaluate a certain system configuration (i.e., an orchestration of assets) with respect to its E2E trustworthiness before deploying it.

By facilitating the generation of trustworthiness values of different granularity, our approach supports the decision-making process in the design phase. In other words, the designer is able to perform "what-if" scenarios and adjust the system structure and redundancy levels in order to meet her goals.

As an initial step, the designer selects the evaluation criteria to be used, i.e., the weights of relevant trustworthiness attributes with respect to the overall E2E trustworthiness of the complete system. The weights represent the preferences regarding the relevance of each attribute, and can be specified e.g. as percentage values. For the Fall Management System, the following list of attributes and associated weights is specified: Privacy (40%), Availability (100%), Reliability (80%), Response Time (100%), Learnability (40%), Effectiveness (60%) and Functional Correctness (60%). The definitions of these trustworthiness attributes are given in [5].

Then the designer creates a set of workflow graphs, each representing a certain feature or usage scenario of the system. The workflows are based on the system model shown in Fig. 4. She has the flexibility to exclude some asset categories that are less important, or make assumptions about the trustworthiness of the relevant asset instances. Even though for example humans play a key role in STS, the workflow concept allows us to focus only on certified software assets that are available on a marketplace. The resulting two workflow graphs for the Fall Management System are shown in Fig. 5.

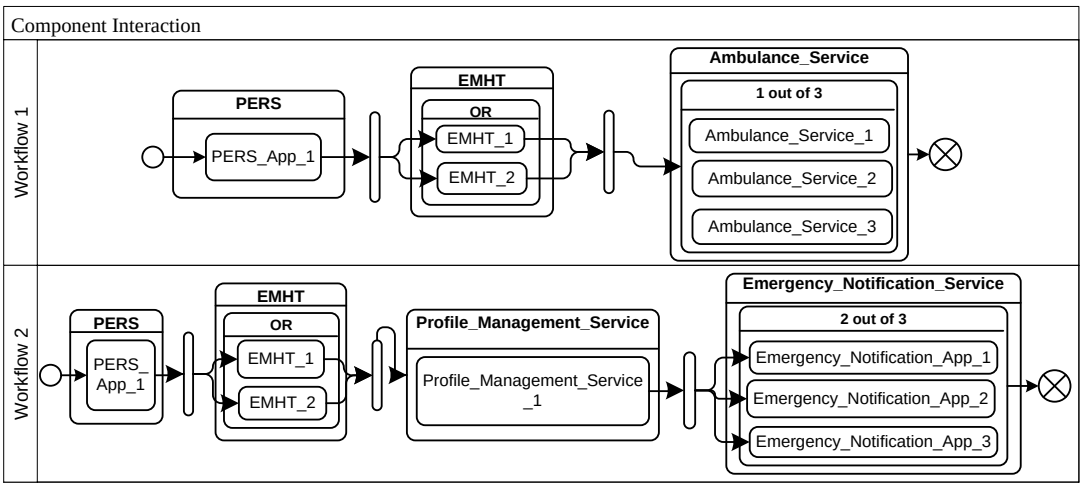

Fig. 5. Workflow Graphs of the Fall Management System.

Workflow 1 consists of three asset categories defined in the abstract system model shown in Fig. 4. For each one the designer has selected concrete asset instances that are available as implementations or realizations of the assets, as well as their redundancy relations (i.e., cardinality and redundancy type). In our scenario, the selected system composition consists of a single instance of the PERS device, i.e., "PERS_App_1" which is a PERS implementation for mobile phones, two EMHT instances, including the main "EMHT_1" and a backup "EMHT_2" (indicated by the "OR" type in the graph), and a pool of three Ambulance Service instances, one of which should be available at a certain point in time (denoted by the "1

out of 3" redundancy type). These asset instances or instance groups are modeled in a sequential order, indicated by the directed edges between them. The asset instances involved in a workflow are part of a certain usage scenario of the system that is to be developed. In particular, this workflow specifies that the "PERS_App_1" has some functional dependency with either "EMHT_1" or "EMHT_2", which in turn calls one out of three Ambulance Services. Thereby some overall system functionality is provided to the user, e.g., the request of an ambulance as a reaction to an emergency alarm handled by the EMHT.

The second workflow graph shown in Fig. 5 describes another scenario or functionality, which should also be provided by the Fall Management System.

It involves a slightly different set of asset categories that are used to notify relatives in an emergency situation.

Again, we focus on software assets, so the notification is performed by Emergency Notification Apps in this workflow. We assume that at least two out of three relatives should be informed in this concrete case, so that a minimum of two respective applications are involved in successfully carrying out the scenario described in this workflow.

The next step includes providing a trustworthiness certificate for each asset instance that appears in the workflows. Certificates carry the trustworthiness metric values that have been approved by some certification authority, and are used as the basis for E2E trustworthiness calculation. Then, as depicted in Fig. 3, the first level or step of trustworthiness computation consists of creating an E2E formula for each trustworthiness metric and workflow. Based on the metric type and metric interpretation, as well as information about the involved component's interactions present in the workflow graphs, an E2E formula for each trustworthiness metric and workflow is created.

For instance, the EMHT instances are modeled as an AND (or "multi-choice and multi-merge") structure, while the Ambulance Service instances and the respective group have "k out of n" semantics. In case of Workflow 1, the following E2E formula will be created for multiplicative metrics that have a positive interpretation (higher values being desirable):

$$e_{\mathrm{Workflow_x,m}} = \prod_{i=1}^{n} a_{\mathrm{i,m}} = tw_{\mathrm{PERS_{App_1},m}} * \Big[[1 - (1 - tw_{\mathrm{EMHT_1,m}})(1 - tw_{\mathrm{EMHT_2,m}})] * [1 - (1 - tw_{\mathrm{Amb_{Serv_1},m}})(1 - tw_{\mathrm{Amb_{Serv_2},m}})(1 - tw_{\mathrm{Amb_{Serv_3},m}})\Big]$$

Where $tw_{i,j}$ is the TW Metric value of the asset instance i for TW metric m.

This formula represents a generic template that needs to be filled with metric values for each involved asset instance. The metric values extracted from certificates are used to calculate separate E2E values for each metric and workflow. In our example, the "mean run-time availability" metric, which belongs to the "availability" attribute, is a multiplicative one, has positive interpretation, and is composed and calculated using formula above, resulting value of approximately 79%. All involved software assets in the system composition, and redundancies are taken into account. As another example for this multiplicative metric, for workflow 2, we get:

$$\begin{aligned} e_{\mathrm{Workflow2,m}} = {} & tw_{\mathrm{PERSApp1,m}} \\ & * \{1 - [(1 - tw_{\mathrm{EMHT1,m}}) * (1 - tw_{\mathrm{EMHT2,m}})]\} \\ & * tw_{\mathrm{ProfileManagementService1,m}} \\ & * \{[(1 - tw_{\mathrm{EmergencyNotificationApp1,m}}) \\ & * tw_{\mathrm{EmergencyNotificationApp2,m}} \\ & * tw_{\mathrm{EmergencyNotificationApp3,m}}] \\ & * [tw_{\mathrm{EmergencyNotificationApp1,m}} \\ & * (1 - tw_{\mathrm{EmergencyNotificationApp2,m}}) \\ & * tw_{\mathrm{EmergencyNotificationApp3,m}}] \\ & + [tw_{\mathrm{EmergencyNotificationApp1,m}} \\ & * tw_{\mathrm{EmergencyNotificationApp2,m}} \\ & * (1 - tw_{\mathrm{EmergencyNotificationApp3,m}})] \\ & + [tw_{\mathrm{EmergencyNotificationApp1,m}} \\ & * tw_{\mathrm{EmergencyNotificationApp2,m}} \\ & * tw_{\mathrm{EmergencyNotificationApp3,m}}]\} \approx 0.3 \end{aligned}$$

Similarly, for each metric, a separate E2E metric will be created, and corresponding trustworthiness values will be calculated respectively. For instance, in the case of concave metrics where lower values are desirable (e.g., "mean error rate" that belongs to the reliability trustworthiness attribute), the following E2E TW formula will be created:

$$\begin{aligned} e_{\mathrm{Workflow1,m}} &= \max(a_{\mathrm{PERS,m}}, a_{\mathrm{EMHT,m}}, a_{\mathrm{Ambulance,m}}) \\ &= \max\begin{pmatrix} tw_{\mathrm{PERS1,m}}, tw_{\mathrm{EMHT1,m}}, tw_{\mathrm{EMHT2,m}}, \\ tw_{AmbServ1,m}, tw_{AmbServ2,m}, tw_{AmbServ3,m} \end{pmatrix} \end{aligned}$$

The formulae for the rest metrics would be created in similar way, according to Table I. Trustworthiness attribute can be quantified by multiple metrics. Hence, the framework presented in Section IV allows for aggregating and calculating E2E metrics on different levels of granularity. The first step is to abstract from metrics and calculate an E2E value on the level of trustworthiness attributes per workflow. More specifically, if multiple metrics characterize an attribute, in this example we use the minimum value per workflow and metric as the E2E value for that attribute and workflow combination. In order to do so, in this case metrics with negative interpretation are transformed into ones with a positive interpretation by taking the residual complementary probability value.

If we consider the trustworthiness attribute availability (identified by attrId) that contains the trustworthiness metrics m1,…, m6 then its value for the workflow 1 is be computed as follows:

$$\begin{aligned} & tw_{Workflow_1,attrId} \\ & = min\ (tw_{Workflow1,m1}, tw_{Workflow1,m2}, tw_{Workflow1,m3}, \\ & \quad tw_{Workflow_1,m4}, tw_{Workflow_1,m5}, tw_{Workflow_1,m6}) \\ & \qquad = min(0.9,0.63,0.385,0.5,0.89,0.48) \\ & \qquad = 0.385 \end{aligned}$$

According to the framework, the next step is to abstract from several workflows, and to compute the overall E2E values of the whole system per trustworthiness attribute. Following a similar "pessimistic" approach, this value is determined accordingly by calculating the minimum value of all workflows. For the attribute "Availability", this will result in the following E2E value:

$$tw_m = \min(tw_{\mathrm{Workflow1,m}}, tw_{\mathrm{Workflow2,m}}) = \min(0.79,0.3) = 0.3$$

Finally, the overall E2E trustworthiness of the whole system has to be calculated, as an abstraction from separate attribute values. To this end, the weights that have been initially assigned to each attribute from the designer are taken into account in order to compute the weighted average. This calculation is reflected in the following formula for our example:

$$\frac{\sum w_{Attr} * tw_{Attr}}{\sum w_{Attr}} = \frac{0.4 * 0.385 + 1 * 0.3 + 0.8 * 0.41 + 1 * 0.34 + 0.4 * 0.5 + 0.6 * 0.4 + 0.6 * 0.37}{0.4 + 1 + 0.8 + 1 + 0.4 + 0.6 + 0.6} \approx 0.3$$

The resulting E2E trustworthiness values on different levels of granularity (i.e., per workflow and metric, per workflow and TW attribute, per TW attribute, as well as one overall TW value) allow the designer to evaluate and document the trustworthiness of different alternative system compositions on the instance level, and consequently helps in making informed design decisions.

VI. Conclusion and Future Work

In this paper, we addressed the problems of the commonly used evaluation techniques and metrics for evaluating E2E trustworthiness. Component-based software development introduces the challenge of considering different component structures for determining an "end-to-end" trustworthiness value based on metrics. The system structure needs to be considered and reflected in the E2E trustworthiness metric that is used to calculate these values. Especially redundancy structures, which are introduced to assure correct system performance and thereby increase trustworthiness levels at design-time, require consideration in E2E trustworthiness calculation. The explicit description of respective metrics is a precondition for the calculation of E2E trustworthiness value, which requires certified metric values of each involved asset as parameters. This evaluation result will be documented and used to support making informed design decisions.

Using Eclipse Process Framework we will describe the process of applying the proposed framework in more detail (cf. [4]). This will provide system designer with guidance information about when and how to evaluate the designed STS in within the development life-cycle, and which work products are expected as outcome of applying the techniques. Furthermore, in order to support the risk-based approach, while the computational approach is performed on application level, relying on measurements of trustworthiness attributes of software asset instances available on the marketplace, the risk-management approach is helpful on the higher level of abstract assets, i.e. asset categories that can be realized by multiple instance implementations. Specifically, at design-time it is essential to identify trustworthiness requirements as controls to prevent threat activity at run-time. We find that using risk analysis in complementary way to commuting approach can characterize the STS to best regarding the trustworthiness threats. The initial steps and concept toward complementing computational evaluation are sketched in our work [28].

References

[1] Sommerville, I.: Software Engineering. 9th Ed., Perarson, 2011.

[2] Whitworth, B.: A Brief Introduction to Sociotechnical Systems, *IGI Global*. Massey University Auckland, New Zealand, 2009.

[3] Paulus, S., Gol Mohammadi, N. and Weyer, T.: Trustworthy software development, *In: Communications and Multimedia Security*, 233-247, 2013.

[4] Gol Mohammadi, N., Bandyszak, T., Paulus, S., Håkon Meland, P., Weyer, T. and Pohl K.: Extending Development Methodologies with Trustworthiness-By-Design for Socio-Technical Systems, *In: Trust and Trustworthy Computing (TRUST)*, 206-207, 2014.

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

[6] Gol Mohammadi, N., Paulus, S., Bishr, M., Metzger, A., Koennecke, H., Hartenstein, S., Weyer, T. and Pohl K.: Trustworthiness Attributes and Metrics for Engineering Trusted Internet-based Software Systems. *In: Cloud Computing and Services Science*, (CLOSER Selected Paper), Communications in Computer and Information Science, 19-35, 2014.

[7] Di Cerbo, F., Bezzi, M., Kaluvuri, M. P., Sabetta, A., Trabelsi, S., Lotz, V.: Towards a Trustworthy Service Marketplace for the Future Internet. *In: F. ´Alvarez et al. (Eds.): FIA* 2012, LNCS 7281, pp. 105–116, 2012.

[8] Lenzini, G., Tokmakoff, A., Muskens, J.: Managing Trustworthiness in Component-based Embedded Systems, *In: J. of Electronic Notes in Theoretical Computer Science*, 179., 143-155, 2007

[9] Raheja, D. G. and Gullo, L. J., Design for Reliability. Wiley, 2012

[10] Ali, M., Sabetta, A., Bezzi, M.: A Marketplace for Business Software with Certified Security Properties. *In: Cyber Security and Privacy Communications*, Computer and Information Science, 105-114, 2013

[11] Kan, S. H., Metrics and Models in Software Quality Engineering. 2nd Eds. Addison-Wesley, 2003.

[12] IEEE: IEEE Standard for a Software Quality Metrics Methodology, IEEE Stdandard 1061-1992, 1993 .

[13] Surridge, M., Nasser, B., Chen, X., Chakravarthy, A., Melas, P.: Run-Time Risk Management in Adaptive ICT Systems, *In: 8th Intl. Conf. on Availability, Reliability and Security (ARES)*, IEEE, 102-110, 2013

[14] Decker, G., Kopp, O., Leymann, F., Pfitzner, K. and Weske M.: Modeling Service Choreographies Using BPMN and BPEL4Chor, In: Proceedings of the 20th Intl. Conf. CAiSE, 79-93, 2008

[15] Khalaf , R., Mukhi, N. and Weerawarana, S.: Service–Oriented Composition in BPEL4WS, *In: 12th Intl. Conf. World Wide Web*, 2003.

[16] Amazon (2014) AWS Marketplace URL: https://aws.amazon.com/marketplace/ (visited on 11/20/2014)

[17] Lotz, V., Di Cerbo, F. Bezzi, M., Kaluvuri, S. P., Sabetta, A. and Trabelsi, S., Security Certification for Service-Based Business Ecosystems, *In: The Computer Journal*. Oxford Journals, 2013.

[18] Klatt, K., Brosch, F., Durdik, Z., Rathfelder, C. , Quality Prediction in Service Composition Frameworks, In: Service-Oriented Computing - ICSOC, 131-146, 2012

[19] Liu,. Z., Liu, T., Cai, L. and Yang, G., Quality Evaluation and Selection Framework of Service Composition Based on Distributed Agents, *In: Proc. of the 5th Intl. Conf. on Next Generation Web Services Practices (NWESP)*, vol. 2., 68-73, 2009.

[20] Elshaafi, H., McGibney, J. and Botvich, D., Trustworthiness monitoring and prediction of composite services, In: IEEE Symposium on Computers and Communications (ISCC), 000580-000587, 2012

[21] Zhao, S., Wu, G., Li, Y. and Yu, K.: A Framework for Trustworthy Web Service Management, *In: Proc. of the 2nd Intl. Symp. on Electronic Commerce and Security (ISECS)*. 479-482, 2009.

[22] Malik, Z. and Bouguettaya, A., RATEWeb: Reputation Assessment for Trust Establishment among Web services, *In: VLDB J*. 18 (4). 885-911, 2009.

[23] Hwang, S.-Y., Wang, H., Tang, J. and Srivastava, J.: A probabilistic approach to modeling and estimating the QoS of web-services-based workflows, *In: J. of Information Sciences* 177 (23). 5484–5503, 2007.

[24] Jaeger, M. C., Rojec-Goldmann, G. and Mühl, G., QoS Aggregation for Web Service Composition using Workflow Patterns, *In: Proc. 8th IEEE Intl. Enterprise Distributed Object Computing Conf*., 149 – 159, 2004

[25] Wang, Z. and Crowcroft, J. , Quality-of-service routing for supporting multimedia applications, *In: IEEE J. on Selected Areas in Communications* 14 (7), 1228-1234, 1996.

[26] Cerbo, F.D., Kaluvuri, S.P., Motte, F., Nasser, B. Chen, W.X., Short, S.: Towards a Linked Data Vocabulary for the Certification of Software Properties. *In: 10th Intl. Conf. on Signal-Image Technology & Internet-Based Systems*, pp. 721-727, 2014.

[27] Almerhag, I. A., Goweder, A. M., Almarimi, A. and Elbekai, A. A.: Network Security for QoS Routing Metrics, In: *Intl. Conf. on Computer and Communication Engineering* (ICCCE), 1 – 6, 2010.

[28] Gol Mohammadi, N., Bandyszak, T., Goldsteen, A., Kalogiros, C., Weyer, T., Moffie, M., Nasser, B., Surridge, M.: Combining Risk Management and Computational Approaches for Trustworthiness Evaluation of Socio-Technical Systems, *to appear in CAiSE Forum*, 2015.