Document Type: Deliverable
Title: Novel Trust Metrics
Work Package: WP5
Deliverable Nr: D5.3
Dissemination: Public
Preparation Date: 31 December, 2011
Version: 2.0

Legal Notice
All information included in this document is subject to change without notice. The Members of the TAS3 Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the TAS3 Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.
The TAS³ Consortium

<table>
<thead>
<tr>
<th>Benef. Nr</th>
<th>Beneficiary name</th>
<th>Benef. short name</th>
<th>Country</th>
<th>Date enter project</th>
<th>Date exit project</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Katholieke Universiteit Leuven (Proj. Coord)</td>
<td>KUL</td>
<td>BE</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>2</td>
<td>Synergetics nv/sa</td>
<td>SYN</td>
<td>BE</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>3</td>
<td>University of Kent</td>
<td>KENT</td>
<td>UK</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>4</td>
<td>University of Karlsruhe</td>
<td>KARL</td>
<td>DE</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>5</td>
<td>Technical University Eindhoven</td>
<td>TU/E</td>
<td>NL</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>6</td>
<td>Consiglio Nazionale delle Ricerche</td>
<td>CNR</td>
<td>IT</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>7</td>
<td>University of Koblenz-Landau</td>
<td>UNIKOLD</td>
<td>DE</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>8</td>
<td>Vrije Universiteit Brussel</td>
<td>VUB</td>
<td>BE</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>9</td>
<td>University of Zaragoza</td>
<td>UNIZAR</td>
<td>ES</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>10</td>
<td>University of Nottingham</td>
<td>NOT</td>
<td>UK</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>11</td>
<td>SAP research (S&amp;T Coord.)</td>
<td>SAP</td>
<td>DE</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>12</td>
<td>Eifel asbl</td>
<td>EIF</td>
<td>FR</td>
<td>M1</td>
<td>M50</td>
</tr>
<tr>
<td>13</td>
<td>Intalio Ltd</td>
<td>INT</td>
<td>UK</td>
<td>M1</td>
<td>M18</td>
</tr>
<tr>
<td>14</td>
<td>Risaris Ltd</td>
<td>RIS</td>
<td>IR</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>15</td>
<td>Kenteq</td>
<td>KETQ</td>
<td>NL</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>16</td>
<td>Oracle</td>
<td>ORACLE</td>
<td>NL</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>17</td>
<td>Custodix nv/sa</td>
<td>CUS</td>
<td>BE</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>18</td>
<td>Medisoft bv</td>
<td>MEDI</td>
<td>NL</td>
<td>M1</td>
<td>M12</td>
</tr>
<tr>
<td>19</td>
<td>Karlsruhe Institute of Technology (KIT)</td>
<td>KARL</td>
<td>DE</td>
<td>M1</td>
<td>M48</td>
</tr>
<tr>
<td>20</td>
<td>Symlabs SA</td>
<td>SYM</td>
<td>PT</td>
<td>M18</td>
<td>M30</td>
</tr>
</tbody>
</table>

Contributors

<table>
<thead>
<tr>
<th>Name</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Slim Trabelsi (Editor)</td>
<td>SAP</td>
</tr>
<tr>
<td>2 Gilles Montagnon</td>
<td>SAP</td>
</tr>
<tr>
<td>3 Michele Bezzi</td>
<td>SAP</td>
</tr>
<tr>
<td>4 Jerry den Hartog</td>
<td>TUE</td>
</tr>
<tr>
<td>5 Nicola Zannone</td>
<td>TUE</td>
</tr>
<tr>
<td>6 Daniel Trivellato</td>
<td>TUE</td>
</tr>
<tr>
<td>7 Antonia Bertolino</td>
<td>CNR</td>
</tr>
<tr>
<td>8 Eda Marchetti</td>
<td>CNR</td>
</tr>
<tr>
<td>9 Christian Hütter</td>
<td>KARL</td>
</tr>
<tr>
<td>10 Klemens Böhm</td>
<td>KARL</td>
</tr>
<tr>
<td>11 Jens Müller</td>
<td>KARL</td>
</tr>
<tr>
<td>12 Thorsten Haberecht</td>
<td>KARL</td>
</tr>
</tbody>
</table>
Contents

1 EXECUTIVE SUMMARY ............................................................... 5
2 TABLE OF ACRONYMS ................................................................ 6
3 INTRODUCTION .......................................................................... 7
   3.1 MOTIVATION / CONTEXT ......................................................... 7
   3.2 OBJECTIVE OF THIS REPORT .................................................. 7
   3.3 METHODOLOGY ...................................................................... 8
   3.4 READING GUIDELINE .............................................................. 8
4 RELATED WORK ...................................................................... 10
5 KPI TRUST METRIC ................................................................. 12
   5.1 KPI-BASED TRUST MODEL ..................................................... 12
   5.2 KPI TRUST POLICY LANGUAGE .............................................. 13
   5.3 KPI MODEL FOR A FEDERATION OF TRUST ............................ 14
6 TESTING-BASED TRUST METRIC ............................................. 15
   6.1 MOTIVATION .......................................................................... 15
   6.2 ONLINE COMPLIANCE TESTING ENGINE ............................... 16
   6.3 USING THE KPI-BASED TRUST MODEL WITH OCT .................. 18
7 CREDENTIAL BASED TRUST METRIC ....................................... 22
   7.1 MOTIVATION .......................................................................... 22
   7.2 CREDENTIAL-BASED TRUST POLICY AND TRUST METRIC LANGUAGES ... 23
   7.3 CTM SERVICE ARCHITECTURE AND IMPLEMENTATION .......... 25
8 PRIVACY-PRESERVING TRUST METRIC ................................... 27
   8.1 PRIVACY-PRESERVING TRUST SYSTEM .................................. 27
   8.2 USE CASE ............................................................................. 27
      8.2.1 Architecture .................................................................. 28
      8.2.2 Flowchart .................................................................... 29
      8.2.3 Trust Model ................................................................. 30
9 USABILITY OF TRUST POLICIES .............................................. 34
   9.1 POLICY WIZARD .................................................................... 34
      9.1.1 Classification of Trust Policies ........................................... 35
      9.1.2 Designing a Policy Wizard ................................................ 36
      9.1.3 Available Constraints ...................................................... 36
   9.2 TRUST VISUALIZATION .......................................................... 39
      9.2.1 Implementation ............................................................... 40
      9.2.2 Performance Evaluation ................................................... 44
1 Executive Summary

This document presents the research results for the novel trust metrics supported within the TAS³ framework. These metrics try to answer to crucial issues not yet addressed in the literature although their impact in the trust relationship between different entities. Four trust metrics are proposed in this deliverable: the first one relies on the key performance indicators (KPI) parameters to build a flexible and federated trust model in which every user will be able to adapt the trust metric to his proper trust objectives. The second trust metric focuses on the audit logs for security testing systems. We show in this solution how audit data can be analyzed in order to evaluate the trust level of a tested entity. This trust level should reflect the resilience of a system against security breaches. The third trust metric relies on an ontological representation of the credential hierarchy for the evaluation of the credential based trust models. The last trust metric proposes the first trust model based on negative feedbacks in order to evaluate the trustworthiness of an entity dealing with private and personal information. This system is able to track personal data and sanction the entity that is not compliant with his privacy policy. Finally, we present two tools that increase the usability of trust policies. All these solutions represent a real innovation in the trust metric literature and impact the quality of the security functionalities proposed by the TAS³ solutions.
# Table of Acronyms

CTM: Credential-based Trust Management  
BTM: Behavioural Trust Management  
KPI: Key Performance Indicators  
KPITM: Key Performance Indicators Trust Model  
OCT: Online Compliance Testing  
PKI: Public Key Infrastructure  
PPTS: Privacy Preserving Trust System  
SLA: Service Level Agreement
3 Introduction

3.1 Motivation / Context

Quantifying trust is not an easy task, although there are many proposals of trust metrics in the literature, no consensual quantitative trust metric has been proposed so far. The word "metric" refers to the fact that it could be possible to impose a layer of trust over a society and to define a sort of metric space on it. A metric space is a set where a notion of distance (called a metric) between elements of the set is defined. Most of the trust metrics used now are subjective and do not offer the possibility to justify or adapt the metric to a particular context. Many questions related to the metric choice and the rating systems still remains not addressed.

A trust decision can be a transitive process, where trusting one piece of information or information source requires trusting another reference or associated source. For example, one might trust a book and its author because of the publisher, and the publisher may be trusted only because of the recommendation of a friend. Winslett's [1] work in policy-based trust call it “credential chains” (the issuer of one credential is the subject of another), the majority of transitive trust computation has been focused on using reputation. A key recent example of this approach is proposed by Golbeck and Hendler [2][3], which describe how trust is computed for the application TrustMail. Reputation is defined as a measure of trust, and each entity maintains reputation information on other entities, thus creating a “web”, that is called a web of trust. The quantification of this trust and associated algorithms are called trust metrics.

A considerable amount of research has been carried on recently on these and related topics, such as trust metrics [4, 5, 6, 7, 8], reputation systems [12], and personalizing PageRank [9]. However most of the current research takes the assumption that every user has an objective trustworthiness value and the goal of the techniques is just to guess this correct value. Conversely, we think that such an assumption is misleading. We argue that these techniques should take into account the fact that different users can have different opinions about a specific user or behaviour.

3.2 Objective of this Report

The focus of this research report is to come up with new trust metrics that enhance the current state of the art by addressing uncovered issues related to the trust modelling. This study aims to stick as much as possible to the reality in order to detect new parameters related to the user behaviour that can severely impact the trust evaluation. The new trust metrics exposed in this document try to propose an understandable metrics space reflecting the perceived consequences of the events that happen during an interaction between TAS³ users. This theoretical research work open new perspectives in the domain of trust metrics and try to bring innovative solutions to the TAS³ project.
3.3 Methodology

The work presented in this document follows three main research methodologies in order to propose new trust metrics: (1) Taking advantage from what we already done and enhance it (2) Aggregating the solutions from different work packages and build a unified system (3) Emerging with a completely new solution.

Starting from what we proposed in [D5.1] and [D5.2] we show in this deliverable how KPI and Credential models can answer to some new trust challenges through specific novel and quantifiable trust metrics application domains. We also propose to take benefit from the results of the Online Compliance Testing Engine developed in the WP10 [D10.2]. The bridge made between the work done in WP5 and WP10 has two main impacts: the first has a research value, in the sense that the combination of a performance based trust metric and a security auditing tool results of an innovative trust metric. The second impact tend to demonstrate how the different components of the global TAS³ framework can interact and collaborate harmoniously in order to come out with an efficient security solution. Scaling the privacy leakage phenomena into a trust metric was a completely new problematic to address, for this reason we propose a completely new trust model that takes into account only negative feedbacks (related to privacy violations).

In [D5.2], a trust policy language for behavioural trust management has been defined. Since trust is a very subjective issue, each user has individual policies on how to derive trust from feedback. Our trust policy language is flexible enough to support such subjective trust policies. It provides customizable calculation rules to aggregate feedback into reputation values. This approach enables users to define trust policies that meet their individual notion of trust. However, we cannot assume that the users are familiar with the formulation of trust policies. In particular, we cannot suppose that they learn our trust policy language. Thus, we have to assist users with the formulation of trust policies and help them to understand the effects of their policies. We present two tools that increase the usability of trust policies.

In this document, we use different types of UML [38] and FMC [37] diagrams to specify the proposed architectures. For interaction diagrams, we use sequence diagrams to show how components communicate with each other. For examples and informal descriptions we also use other illustration formats.

3.4 Reading Guideline

This document is structured as follows. In Section 4 we start with an overview and a classification of the trust metrics solutions in the literature. Section 5 introduces the KPI-based trust metric that takes into account the user objectives in order to provide an adaptative perception from the performance values of an entity. Section 6 presents a hybrid system that allies the KPITM with a

---

compliance testing tool (OCT) in order to provide an audit based trust metric. In Section 7, we describe solutions which support the use of ontologies for interoperability in order to scale credential based trust models into a new metric. Section 8 describes a new privacy preserving trust metric that evaluates the impact of private data leakage. This trust metric evaluates the compliance of an entity collecting personal data with their privacy policies. Finally, Section 9 presents the tools we have developed to increase the usability of trust policies.
4 Related Work

First proposals for trust metrics appeared early nineties, where trust metrics have been deployed in various projects to support the Public Key Infrastructure [15]. Metrics proposed in [16], [17], [18], and [19] count among the most popular ones for public key authentication. Trust metrics were not dedicated to PKI based solutions but other applications raised new issues not yet covered by the early models. Systems P2P networks, ubiquitous, mobile computing, and rating systems for online communities, where maintenance of explicit certification authorities is not feasible anymore, have raised the research interest in trust. Ziegler and Lausen [13] identified three principal dimensions with distinctive features (see Figure 1). These axes are not orthogonal, though, for various features impose restrictions on the feature range of other dimensions. For instance, [20] differentiates between local and global trust, and distinctive features between scalar and group trust metrics are discussed in [16].

![Figure 1: Trust metrics classification](image)

Trust metrics may basically be subdivided into ones with global, and ones with local scope. Global trust metrics take into account all peers and trust links connecting them. Global trust ranks are assigned to an individual based upon complete trust graph information. Many global trust metrics, such as those presented in [22], [20], and [21], borrow their ideas from the renowned PageRank algorithm [9] to compute web page reputation. The basic intuition behind the approach is that nodes should be ranked higher the better the rank of nodes pointing to them. Obviously, latter approach works for trust and page reputation likewise. Trust metrics with local scope, on the other hand, take into account personal behaviour. Some studies claim that only local trust metrics are “true” trust metrics, since global ones compute overall reputation rather than personalized trust [23]. Local trust metrics take the agent for whom to compute trust as an additional input parameter and are able to operate on partial trust graph information. The rationale behind local trust metrics is that persons an agent x trusts may be completely different from the range of individuals that
agent $y$ deems trustworthy. Local trust metrics exploit structural information defined by personalized webs of trust. Hereby, the personal web of trust for individual $x$ is given through the set of trust relationships emanating from $x$ and passing through nodes it trusts either directly or indirectly, as well as the set of nodes reachable through latter relationships. Merging all webs of trust engenders the global trust graph. Local trust metrics comprise Levien’s Advogato trust metric [24], metrics for modelling the Public Key Infrastructure [17,18,19], Golbeck’s metrics for Semantic Web trust [25], and Sun Microsystem’s Poblano [26].

In this deliverable we propose both new local and global trust metrics. The KPI and audit based trust metrics are local in the sense where the perception of the trustworthiness is adaptable to all the users according to their trust criteria. The privacy preserving trust model is more a global trust metric where the ratings are all the same for all the users and trust value is calculated in a centralized manner. The same perception of trust is shared by all the users.
5 KPI Trust Metric

Key Performance Indicators (KPI) are financial or non-financial measures or metrics used to measure progress, performance or goals of an organization and provide another source of trust feedback. KPI trust management (KPITM) [D5.1] is an approach in which trust evaluation is based on the experience on KPIs shared by different users.

5.1 KPI-based Trust model

We propose a KPI-based trust model that takes into account trust metrics based on KPI. In our approach, a user can express his trust preferences via a policy that specifies the sources of the performance indicators factors and how the factors should be combined to obtain a reputation score. According to his business objectives, the user is able to prioritize some indicators by setting a strong weight affecting the result of the trust score. These indicator values are then aggregated and normalized in order to obtain a unified reputation value.

Compared to the traditional trustworthiness models deployed in famous internet websites, such as Amazon or E-Bay, that rely on a subjective rating system, we propose a trust model based on quantitative facts (measurable performance). In the KPI-based model, each item has a semantic meaning that explains the context of the measured value related to every performance parameter (e.g., delivery time, financial results, overall budget, company growth, etc.). Combining different KPI items offers the possibility to the trustee to customize his trust evaluation by expressing complex queries.

This new model adds more dynamicty to the trust evaluation due to the freshness and the diversity of the KPI values that are evolving continuously in time. An entity that wants to connect to a KPITM system in order to evaluate the trust of another entity has to select three elements: The KPI items that are relevant for him, the URI or the pointer to the source of the performance value, and finally, the ratios affected to every item in order to prioritize some values during the trust evaluation. All this information must be contained in the core query sent to the KPITM engine that will automatically connect to the different sources, get the values of each item and compute the trust value. More details about this trust model are available in [D5.1].

The KPI-based trust model offers the possibility to quantify the trustworthiness values according to business objectives and permit to any business process component to determine which business partner is more trustable according to an objective estimation. In particular, our KPI-based trust model allows a trustee to evaluate the weight of a recommendation by applying the business objective scale of the recommender.

More formally, the KPI-based trust model that we present in this document is composed of three complementary layers:

- Performance Indicator Values: are collected from the different sources providing the values related to the performance items.
• Business Objectives Scale: are fixed by the trustee according to the performance indicators related to their business objectives. An interval of values (min and max) must be chosen for every performance indicator in order to normalize the measured value with a [0,1] scale. The [0,1] normalization rule is written as follows:

\[
\begin{align*}
1 & \quad \text{if } K_i > K_{\text{max}} \\
0 & \quad \text{if } K_i < K_{\text{min}} \\
\frac{K_i - K_{\text{min}}}{K_{\text{max}} - K_{\text{min}}} & \quad \text{otherwise}
\end{align*}
\]

Higher is better KPI normalization  Lower is better KPI normalization

Where \( K_i \) is the measured performance indicator value, \( K_{\text{min}} \) and \( K_{\text{max}} \) are the maximum and minimum values declared in the business objectives scale. If lower values are better (e.g. the delivery time example) the value is reversed by subtracting it from 1.

• Trust Level Value: is the aggregation of all the normalized performance indicators plus eventually some external values like the recommendation from other trusted entities.

5.2 KPI Trust Policy Language

The KPI trust service is a modular component that can be invoked by means of KPI estimation requests. Request are specified as SQL-like queries. They contain the following structured elements describing the input/output parameters of the request that are transmitted to the KPI engine.

• Entity: if the KPI evaluation request is related to more than one business entity, we can number and distinguish them using the entity element.

• \( \text{kpiTotal} \): represents the KPI-based trust value related to one entity.

• Name: name of the performance indicator

• Type: type of the performance indicator

• Url: is the reference about the location of the performance indicator provider (this element could be different from an URL, like a point for a DB connector).

• Scale: the scale proposed by the user in order to normalize the performance indicator value

• Weight: in order to prioritize the performance indicators a weight is used to influence the kpiTotal computing

• \( \text{kpiFactor} \): is the value of one trust indicator

Queries are structured as follows:
GET set/subset of results

FROM disjunction=conjunction of KPI Elements

WHERE sequence of Parameters

With:

• Results: contains the elements kpiTotal, KpiFactor
• KPI elements: contains the elements Entity, Name, Type, Url
• Parameters: contains the elements Scale, Weight

For example a manager in a research team that is wondering to hire internally a researcher can send this query to the local HR database:

GET kpiTotal

FROM URL.User1Filed_Patents.US AND

URL.User1.Published_Papers AND

URL.User1.Research_Projects

WHERE URL.User1Filed_Patents.US ∈ [1; 5] | | 0,5 AND
URL.User1.Published_Papers ∈ [10; 20] | | 0,3 AND
URL.User1.Research_Projects ∈ [1; 12] | | 0,2

5.3 KPI Model for a Federation of Trust

The KPI based trust model makes the trust customizable and portable. This new trust metric adds more objectivity and flexibility by providing to the user the possibility to adapt the trust evaluation to his own objectives and match with his personal perception of trust. This approach make easy the federation of trust; when different websites wants to federate their trust and reputation system, the KPI-based trust model provides the mean to adapt a trust evaluation to any kind of local perception, as identity federation systems can do [10]. Actually there are no real trust federation mechanisms; for example we cannot reuse an Amazon reputation profile in Ebay. If a user wants to open a virtual store in Ebay, knowing that he already has a very good reputation in Amazon, he is obliged to setup a new reputation profile. The KPI-based trust model proposes to adapt the trust and reputation perception if the concerned entities share the same KPI's (like for example online-stores), in this case it will be easy to share a reputation between different independent domains.
6 Testing-based Trust Metric

6.1 Motivation

One of the objectives of this document is to propose a trust metric on logging and audit results. In fact, audits and logs can reflect the behavior of a system or an entity if we are able to understand and analyze correctly the events stored in such reports. The association of a trust level to an event stored in a log file can, for instance, be done by combining an objective and a subjective measure, i.e., an analysis of the data in the event log and the confidence we have in the result of one operation. Security logs [11] of user behavior not only record the past and present of system, but also can detect the tampering behavior. In general, the security log has the features include: data access control, which means that the contents of the security log should correspond with the user's behavior; verifiable, there are some simple and effective way to verify all entries; tamper-resistant, except the founder.

In this document we propose to rely on the KPI-based trust model in order to analyze the results of the on-line compliance testing process and come-up with a trust indicators. In particular, we propose to implement some of the trust-based selection criteria of services within the KPITM also combining indicators from OCT [D10.2]. Our approach aggregate the functionalities of three existing components from the overall TAS³ architecture (Figure 2).

![Figure 2: TAS³ components architecture](D2.1)
6.2 Online Compliance Testing Engine

The main idea of the OCT engine [D10.2] is that services are submitted to the execution of a set of test cases in their real execution environment, while they possibly interact with other services (and may even be unaware that an invocation they receive comes from OCT). OCT validation may be performed periodically or after some relevant event, aiming at verifying that every service in the TAS³ federation abides by the manifested policies and/or complies with the functional specifications. For example the OCT engine can be used to test the resilience of an access control system to a server; a set of access requests are executed with different roles and over different resources and if an authorized/unauthorized access are allowed, the security breach will be reported in the audit log. The test strategy selection is an orthogonal aspect to the on-line compliance testing infrastructure implementation and deals with the selection of a set of meaningful test cases to be executed. The OCT engine provides two main functionalities:

- **The UML-based tool**: that is embedded in the OCT infrastructure. This tool allows the tester to specify the abstract test cases that will be automatically translated into the executable ones by Model-driven technologies.

- **The X-Create framework**: that is an original test strategy and the description of a framework for supporting it, called XaCml REquests derivAtion for TEsting (X-CREATE). This framework allows the automated derivation of a test suite starting by a XACML 2.0 [38] TAS³ service policy. Differently from the existing solutions the proposed approach exploits the XACML 2.0 Context Schema to derive a universally valid conforming test suite of XACML requests that are customizable by the X-CREATE framework, to any specific TAS³ policy [D2.1].
Figure 3: OCT architecture

The overall architecture of the OCT system (see Figure 3) contains seven five components and two external entities. The internal components are:

- **Test Driver**: is a component that implements the logic of the test planner, supporting the “perpetual” and on-line testing of software services. In particular, the Test Driver includes a test scheduler that activates the on-line testing sessions either in event-driven way or periodically. It can be viewed as the configuration interface used by the administrator in order to schedule the tests.

- **The test cases repository**: it is a dataset that contains the testing cases to be executed. Each test case specifies the service under test, the remote operation to invoke, and the role that the Test Robot has to play.

- **The Oracle**: is the decision maker. The oracle is a reasoning engine that interprets the results of the test cases executed by the Tester Robot. Using a knowledge database the oracle is able to estimate if the tests are successful or not.

- **Tester Robot**: is the core of the testing system. It plays the role of the communication bus between all the components, It aggregates all the information taken from the test case repository and the identity providers, then it performs the security testing within the targeted server, it consults the oracle for the testing decision, and finally stores the results into the audit repository.

- **Test result repository**: is a dataset that contains a structured logging files stating about the results of the role-based access compliance testing.

The two external actors of the system are:

- **The Identity provider**: that provides the credentials and the tokens necessary to perform the access control requests. Since the OCT engine has to behave like a real user, he needs to use the roles of real users, with temporary credentials certifying the attribute roles are necessary to challenge the access control system of the targeted server.

- **The service**: is the target entity to be evaluated by the OCT engine.

The sequence diagram below (Figure 4) describes the execution sequence of the OCT engine. 1 the test scenarios are configured via the tester driver, the test sketches are then sent to the Tester Robot. 2 the Tester Robot loads the test cases corresponding to the sketches. 3 According to the roles required by the test cases, the Tester Robot asks the Id Provider a set of certified roles. 4 the Tester Robot start the execution of the test cases. 5 the results of the tests are then transferred to the Oracle in order to be interpreted and evaluated. 6 after evaluating all the results, the Tester Robot stores the audit report generated by the oracle into the Audit Result Repository.
6.3 Using the KPI-Based Trust Model with OCT

Building a trust model over an audit analysis is a quite new concept that we try to bring in the TAS3 framework. To our knowledge only one tentative was proposed by Goodman and Verbrugge [27] to use a simple trust model for detecting cheaters in P2P games. They rely on a peer auditing provides a simple method for detecting malicious clients; however, in order to determine which, if not both, of the two clients involved in a failed audit is responsible for the suspicious behaviour, it becomes necessary to monitor failed audits. In order to separate cheaters from honest clients a trust metric is implemented. Trust is calculated based on a client’s history of monitor comparisons and quick tests. This approach is quite static and naïve, because it supposes that the input parameters for the trust model are only based on good or bad results reported in the audit. In reality, test result repository generated by OCT contains too much other information that can be used as input parameters for the trust model. We tried to analyze the information contained in the audit log generated by the OCT engine and try to identify all the parameters that can be useful for a trust evaluation. We came up with the list detailed below:

<table>
<thead>
<tr>
<th>KPI Element (ITEMS)</th>
<th>Description</th>
<th>Scale</th>
</tr>
</thead>
</table>

Figure 4: OCT sequence diagram
| Role Coverage | The number of roles (R) over the total number of roles defined within the test scenario that have been used to test the service S | Min: 0 Max: 1 |
| Operation Coverage | The number of operations (OP) over the total number of operations exported by the service S, that have been used to test the service S | Min: 0 Max: 1 |
| Strict Role Coverage | The number of roles (R) over the total number of roles defined within the test scenario that have been used to test the service S on a given operation OP | Min: 0 Max: 1 |
| Strict Operations Coverage | The number of operations (OP) over the total number of operation exported by the service S, that have been used to test the service S on a given role R | Min: 0 Max: 1 |
| Service Coverage | The number of pairs (R, OP) over the total number of pairs (R x OP) within the test scenario that have been used to test the service S. | Min: 0 Max: 1 |
| Revealed Problems | The number of failed test cases | Min: 0 Max: LongInt |
over the total number of the test cases executed for a given service S.

Table 1: Testing results KPIs

<table>
<thead>
<tr>
<th>Testing Parameters</th>
<th>Description</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Between Service Tests</td>
<td>Given service S, it is the time passed from the last test case executed on S.</td>
<td>Min: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Max: LongInt</td>
</tr>
<tr>
<td>Time Between Test Scenarios</td>
<td>The time passed from the end of a run of a test scenario and the beginning of the next run.</td>
<td>Min: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Max: LongInt</td>
</tr>
<tr>
<td>Test scenario aging</td>
<td>The time passed from the instantiation/design of the test scenario.</td>
<td>Min: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Max: LongInt</td>
</tr>
<tr>
<td>Frequency</td>
<td>Is the inverse of the time passed from the beginning of a run of a test scenario and the beginning of the next run.</td>
<td>Min: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Max: Real</td>
</tr>
<tr>
<td>Test Scenario Size</td>
<td>The cardinality of the test suite within a test scenario.</td>
<td>Min: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Max: LongInt</td>
</tr>
<tr>
<td>Test Cases per Service</td>
<td>Given a service S, the number of test cases within the test suite having S as target service.</td>
<td>Min: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Max: LongInt</td>
</tr>
<tr>
<td>Test Date</td>
<td>The date of the given Test</td>
<td>Min: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Max: LongInt</td>
</tr>
</tbody>
</table>

Table 2: Testing Parameters

Where a role R within a service federation, is a representation of an entity using a set of attributes. Such attributes can be either domain specific or not.
• A test case is an interaction between a service client (C) and a target service (T). In particular a test cases is a tuple <C, T, OP, R>, where : OP is an operation exported by the T, and R is the role that C plays in such interaction.

• A test suite is a collection of test cases.

• A test scenario is a test suite where each test case is associated with its expected result.

• The test scenario is periodically executed. At each execution, the coverage metrics on the test suite are newly initialized. Thus, monitoring the executions of the test scenario, the number of the test cases that have been run change (i.e. increasing or decreasing) over the time.

• We make the difference between the quantitative data that can be counted as KPI items and qualitative data that is taken as a parameter configuration data.

According to the KPI query language described in section 5.2 the audit data described previously in Table 1 and Table 2 can be written as follows:

GET set/subset of Services
FROM disjunction/conjunction of KPI Elements
WHERE sequence of Testing Parameters

For example we can ask to compute and compare the trustworthiness values of the services that obtained the highest success rates in the security tests during the 5 last months:

GET Services FROM X <- MIN(Services.Revealed_Problems) WHERE Services.test_date >= 5 months AND 0 <= X <= 5

We can also use the priority functions implemented in the KPITM engine in order to match with the objectives of the requester. For example we can ask and compare the trustworthiness values of the services that obtained the highest success rates and the highest role coverage with a priority of 75% for the role coverage.

GET Services FROM X <- MAX(Services.Role_coverage) AND Y <- MIN(Services.Revealed_Problems) WHERE 20 <= X <= 50 AND X.priority= 75%, 0 <= Y <= 5 AND Y.priority= 25%
7 Credential based Trust Metric

Credential-based trust management (CTM) (see [D5.1, D1.1] for an overview of existing CTM systems), is an approach to distributed access control in which access decisions are based on the attributes of a principal. In this section we describe novel trust metrics which support the use of ontologies for interoperability and describe the trust service that supports these metrics.

7.1 Motivation

The TAS3 architecture supports service cooperation across organizations. CTM supports this cooperation by allowing trust to be built by combining credentials from different parties. Credentials are digital certificates attesting that a certain subject has a certain attribute, and are digitally signed by the certificate issuer to ensure their authenticity and integrity. Every principal has the authority to define credentials; a statement defining a credential (or the conditions in which it is issued) is called an assertion. The set of assertions made by a principal is the CTM policy (or simply policy) of that principal. The key aspect in CTM is delegation of authority: a principal may transfer authority over some attribute to other principals by referring to their credentials in its policy.

Delegation in the main strength but also a main challenge for CTM in distributed systems such as the TAS3 architecture. To be able to use credentials of other parties they will need to be understood; a semantic agreement on the vocabulary of parties must be reached for interoperability to be possible.

Example Consider a scenario in the employability setting as detailed in [D9.1]: a placement provider, ePlace, has the following policy (PP) “Only applicants with a MSc in Business Administration issued by an accredited university can apply to managerial positions.” Suppose that a student, Alice, submits an application for a managerial position and attaches to it a credential issued by the University of Nottingham attesting that she has a MSc in Business Management. Despite the fact that the Alice’s degree clearly fits the profile, ePlace would not accept her application. Indeed, the certificate presented by Alice is different from the one required by ePlace.

Most existing CTM languages implicitly assume a complete agreement among principals on the vocabulary for the specification of credentials [27, 28, 29]. This assumption is however too restrictive for short-term or dynamic collaborations, where reaching a complete and precise semantic agreement on the vocabulary is infeasible or simply would be too costly or take too much time.
We address the problem of semantic agreement in two steps: (1) we combine distributed access control with ontologies to give a precise semantics to policies, and (2) we employ the notion of similarity for semantic alignment. Ontologies are increasingly seen as a means for enabling interoperability in heterogeneous systems [30]: augmenting trust management with ontologies ensures mutual understanding among principals [31]. Similarity expresses the degree of semantic resemblance between two ontology concepts [32]. Combining ontologies with similarity allows principals to specify policies in terms of their local vocabulary, increasing their autonomy, while guaranteeing interoperability [33].

7.2 Credential-based Trust Policy and Trust Metric Languages

Trust metrics are queries on trust worthiness that can be presented to the credential based trust service while trust policies describe the configuration of this service. We present a policy specification language based on constraint Datalog, which has been proposed as a foundation for CTM languages [34]. A CTM policy consists of a set of Horn clauses of the form $H \leftarrow B_1; \ldots ; B_n; c$, where $H$ is an atom called head, and body $B_1; \ldots ; B_n; c$, where $B_1; \ldots ; B_n$ are 0 or more atoms, and $c$ is a constraint. CTM policies are specified using three types of constructs:

- **Ontology atoms**: are used to query the knowledge base represented by ontologies; they have the form conceptURI (a) or relationshipURI (a1; a2),
where conceptURI and relationshipURI identify a concept and a relationship in the ontology, respectively. conceptURI (a) holds if a is an instance of conceptURI . relationshipURI (a1; a2) holds if instance a1 is related to instance a2 via relationshipURI .

- **Credential atoms**: have the form cred(issuer; attribute; subject) and represent digitally signed certificates released by an issuer attesting an attribute of a subject. issuer and subject are identified by a unique name (e.g., a public key) and attribute is a conceptURI .

- **Constraints**: are conjunctions of Boolean expressions of the form X ~ Y, where ~ is a comparison operator (e.g., =, >, <, etc.) and X and Y are terms.

```
1  cred(ePlace, ep:validCandidate, X) ← ep:accredited(Y), cred(Y, ep:MScBA, X)
2  cred(UoN, uon:MScBM, Alice)
3  sim(ePlace, ep:MScBA, uon:MScBM, 0.9)
4  cred(ePlace, ep:validCandidate, X) ← ep:accredited(Y), cred(Y, Z, X),
    similar(Z, ep:MScBA) ≥ 0.8
```

**Figure 6: Example Credential based Trust Policy**

Clauses 1 and 2 in **Error! Reference source not found.** represent, respectively, policy PP and Alice's credential. The credential attributes in both clauses and the first atom in the body of clause 1 are ontology concepts. In particular, valideCandidate, accredited, and MScBA are concepts defined in ePlace's ontology (ep), and MScBM is defined in the ontology of the University of Nottingham (uon). Following the XML convention, we use a prefix to denote namespaces (i.e., URIs).

Similarity statements are asserted in the form of credentials. In the language, they are represented by similarity credential atoms of the form sim(issuer, att1, att2, degree), where issuer is the unique name of the principal issuing (and signing) the similarity credential, att1 and att2 are conceptURIs, and degree is the degree of similarity (with value in the range [0...1]) between att1 and att2 in the view of issuer. To increase the flexibility of her policy and enhance interoperability with other principals, a principal might accept credentials about an attribute that is similar to a given known attribute for at least a certain degree. This is expressed by means of the following similarity constraint:

```
similar(att1; att2) ≥ threshold
```

where threshold is the minimum degree of similarity between attributes att1 and att2 required by the principal. This constraint can be used as any other constraint in clauses.

Clause 4 in **Error! Reference source not found.** shows how clause 1 can be modified in order to increase the flexibility of ePlace's policy while preserving its semantics. Now, when ePlace receives credential cred(UoN; uon:MScBM; Alice) from Alice, it checks whether according to the available similarity statements the similarity degree between attributes uon:MScBM and ep:MScBA is higher than
the threshold. Given the similarity credential in clause 3, Alice's application is accepted by the system.

**Definition** The trust metric language consists of the following three types of trust queries:

1. *ground queries*: request a credential issued by the query's recipient attesting that a certain subject has a certain attribute;
2. *attribute queries*: request all the credentials of a given subject (i.e., all his certified attributes) issued by the recipient;
3. *subject queries*: request all the credentials for a given attribute (i.e., the set of subjects having that attribute) issued by the recipient.

### 7.3 CTM Service Architecture and Implementation

Every principal runs an instance of the CTM service and communicates with other principals by exchanging queries and responses. The architecture of the CTM service is based on a PEP-PDP structure fitting well within the TAS³ framework. A high level overview of this architecture of the CTM service is presented in Figure 7: CTM Service Architecture. Trust metric evaluation requests sent to a principal are intercepted by her Policy Enforcement Point (PEP).

Upon intercepting a query, the PEP forwards it to the Policy Decision Point (PDP). The PDP is the component responsible to make a decision based on the policy defined by the query's recipient. When evaluating a query, the PDP can interface with (local and/or remote) ontologies to retrieve information from the knowledge base. The decision of the PDP is returned to the PEP which enforces it, possibly releasing the requested credential. Notice that, since clauses may refer to credentials issued by other principals, the decision of the PDP may in turn be a query for such credentials. Several algorithms have been proposed to deal with the evaluation of distributed policies [27, 28, 35, 36]. The proposed architecture is flexible enough to implement any of them.

Similarity constraints in the policy are resolved by contacting the Ontology Alignment Module. This module computes the degree of similarity between the two attributes in the constraint based on the similarity credentials available in the Similarity Credential Repository.

We implemented a prototype of the PDP in SWI-Prolog. The main reason for using SWI-Prolog is that it provides an interface with ontologies, through the Semantic Web Library. This library consists of packages for reading, querying
and storing RDF documents, and hence ontologies (every ontology can be represented as a set of RDF triples). For example, the ontology atom `ep:accredited('UoN')` can be expressed using the built-in predicate `rdf(UoN, rdf:type, accredited)`, where `rdf:` is the prefix of the URI where relationship `type` is defined. The Ontology Alignment Module is a Python server that communicates with the PDP by exchanging HTTP messages, and the Similarity Credential Repository is implemented in MySQL. In the prototype, we employ the RT algorithm [34] for the evaluation of distributed policies.
8 Privacy-preserving Trust Metric

8.1 Privacy-Preserving Trust System

TAS3 will provide secured transactions and will allow user’s information to be sent to a service provider while respecting user’s privacy policies. However, for many reasons, user’s information may leak from a service provider to the internet without any TAS3 transaction (so not logged nor audited) and without any user consent: policies disrespectful systems, system security weaknesses, hacking... Once user’s information leaked over the internet, there is mostly no way to identify the source of the disclosure. As a trustworthiness indicator, a Privacy-Preserving Trust Metric can be determined by detecting private information leakage and then to link this leak to the original source container (service provider).

We propose this new trust metric based on a reputation system for preventing user’s private data misusage. This, so called, Privacy-Preserving Trust System (PPTS) offers the possibility for any user to track his personal information during the usage by a third party, and to punish any misbehaviour by assigning a bad score into the reputation system. When a leak of private data is detected, by any means, the system identifies the original container, decreases its reputation level and changes the trust metric.

The system uses a novel trust model that relies on negative feedbacks. At runtime, 2 major phases:

- During the service provider discovery, the trust metric provided by the PPTS can be used for the selection (trust based web service discovery);

- At any point in time after the transaction ended, a misuse of user’s data sent to a service provider can be reported. The report assesses that a user private data has been used without user’s consent thus breaking privacy policy.

8.2 Use case

Coming from TAS3 employability scenario (eg: D1.4-APL,[ ref to put here]), an employee should provide private information to employability providers. Information like e-mail address is one of this usual information that user provides in such scenario. Some of these employability providers could illegally re-use addresses for marketing or whatever purpose, exposing the user to the SPAM threat. Corporate or private anti-spam solutions offer a posteriori method to detect and intercept them but the problem is that the user is not able to verify the origin of this data leak. To mitigate this issue it’s possible to create multiple disposable addresses then delete the one that receives too many SPAMs. This mitigation is very limited and doesn’t bring any useful feedback for the corporate and internet community.

As Novel of Trust scenario that can be extended to TAS3 scenarios, we will define the Privacy-Preserving Trust System to identify websites (as service provider) from which stored user’s emails have been used for spam. In this particular scenario identifying original sources of email leak may reduce significantly the
number of junk emails sent, although it can be extended to many other data types.

For this scenario, the PPTS detects websites that provide, intentionally (break of policies) or not, its customer’s email to other 3rd Party. Generally, this leak of information leads to spam and junk emails. Using this system, each time a junk email is received, the website where the email was originally stored is identified. Its reputation is then computed, and helps further users in providing or not their email to this website.

In this particular scenario, the Privacy Preserving Trust System will store and manage the following data:

- the reputation (percentage of trust from 0 to 1) is given for the web site root url: 
  \[
  \text{[websiteURL]} = \text{[0.00 – 1.00]}
  \]
  eg : www.example.com = 0.95

- the necessary bindings for this particular scenario would have the following format : 
  email#[userId]#[websiteURL] <- bind to -> GUID@email.com

eg : email#URI:1156434196#www.example.com <-> 120ba704-a975-4650-9d0f-47387a2ac329@sap.com

### 8.2.1 Architecture

Following FMC notation, this high level architecture could be proposed.

![Privacy Preserving Trust System Architecture](image)

The Privacy-Preserving Trust System is composed of 2 main agents:

1) Reputation Engine
a. store and provide data-collectors reputation (websites in the diagram)
b. get and manage abuse reports
c. compute ranking in terms of respect of privacy policy
d. provide uniquely identified private data to user (from Unique Id Mapper)

2) Unique Id Mapper
   a. map private data to a given data collector
   b. store some context information related to datas (submission date, subdomain,..)
   c. get uniquely identified private data for the mapping

8.2.2 Flowchart

The goal of the Privacy-Preserving Trust System is to rank the reputation (in terms of respect of privacy policy) of any system by tracking the misusage of private data given by users.

![Flowchart Image]

Figure 9: Message sequence flowchart

1st part of the process

- A user wants to enter some private data into a data-collector. For example a form on a website (chosen for the diagram), subscription to web service, any system that may collect user's information.

- The user gets a reputation ranking related to the data collector. This ranking is an indicator about the reliability in terms of respect of privacy policy.

- Assuming the user decides to provide private information, this has to be uniquely identified and linked with the data collector. The ‘Type Based GUID
Generator’ is able to generate, on the fly, a unique identifier as a result of a combination between the user, the private information (can be a picture, video,...) and the data collector (website). It will then attach or embed this ID into the private information. Several technologies exist for this purpose depending on the type of data: DRM, stenography, digital watermarking ... all these technologies will allow retrieving the unique ID later on. By creating this unique ID in such way, the original data collector can be identified wherever the data has been found on the internet.

- The user provides this traceable private data to the data-collector.

- A 3rd party could retrieve this private data. This can be happen intentionally or unintentionally, coming from misusage/loss from the data collector, hacking or non declared commercial agreement.

2nd part of the process

- An unexpected or illegal use of private data is detected
- The user can report this abuse to the PPTS using the detected data.
- PPTS identifies the source of the abuse (website in the diagram)
- PPTS recalculate the ranking of this source.

Once one of your private data is found on the internet without prior agreement, it may be really difficult to know the complete circulation path of the leakage. This solution brings the certitude about the source where your data was placed first. This allows evaluating and ranking data collectors on their privacy protection mechanism and behavior.

You may find a similar user experience for virus detection while browsing the web, but the whole processing and technology needed is different in our invention.

8.2.3 Trust Model

In the system described above only penalization feedbacks are taken into account that makes our trust model quite different from traditional ones. Traditional reputation systems relies on an aggregation of positive and negative feedbacks, ours is only based on negative ones. The first approach to address this issue consist in application of the driving licence penalty points reputation model; this model consists of giving a penalty point each time the target commits an infraction. This approach is quite naïve and unfair in the sense that we cannot reward a good behaviour, and we cannot differentiate a target with a very good behaviour since a long time period and another one with a short time period. Such solution promotes the redemption behaviour; after a long period of time there is no way to differentiate a target that never committed an infraction and a target that committed some infractions in the past.
We propose a new reputation model based on negative feedbacks that take into account the good behaviour of the targets in order to reward the best users. We propose first to reward the honesty increases in the time; longer the honesty period is, higher the reputation will be. This statement permit to differentiate a good reputation acquired from a good behaviour since a long period of time and a good reputation acquired from the bootstrap of the reputation system. The reputation will increase continuously with the time if reputation engine does not receive a negative feedback. This growth can be limited to a threshold value in order to avoid an infinite growth. The reward algorithm follows a logistic growth map and can be written as:

\[ y = r x_t (1 - x_t) \]

\[ y_{\text{max}} = \frac{r - 1}{r} \]

Where \( y \) is the reputation value, \( x_t \) is the growth rate per time unit, \( r \) is related to the maximum threshold value of growth. According to this formula, the evolution of the reputation related to a good behaviour will be like this:

![Figure 10: reputation growth model](image)

In case of negative feedback the reputation value will decrease according to the ratio \( \lambda \) of negative feedbacks over the amount of data collected by a particular target. For example a website that collected 1000 e-mail addresses and displayed 10 will be penalized by -0.01 and the web site that collected 100 e-mails and displayed 50 will be penalized by -0.5. The malicious target can benefit of a redemption period in which he can enhance his reputation by respecting the privacy of the personal data. The evolution of the negative reputation curve will be like this:
In order to catch these two phenomena into a single trust model, we propose to use two reputation zones: the positive and the negative zone. These two zones are spearred by a neutral line that plays the role of bootstrap level for a new target added to the system. When a new target is added to the system, his reputation will increase in the positive zone according to his lifetime. As soon as a negative feedback is recorded, the reputation value will fall below the neutral line with a negative value estimated to $\lambda$. Then the redemption period starts until the next negative feedback (if it happens). The reputation value is calculated then according to the following equation:

$$y = rx_t (1 - x_t) - \left( \frac{r - 1}{2} \right) (1 + \lambda)$$

According to our trust model the evolution of a reputation can behave like the following sample:
Figure 12: Example of trust variation over the time

We can see in the Figure 12 that the reputation of a target starts in the neutral line and raises with the time into the positive reputation zone. As soon as the target an infraction is detected, the reputation value falls down under the neutral line with the ratio $\lambda$ and this can happens each time a negative feedback is detected.
9 Usability of Trust Policies

One approach to build user trust in service providers is Behavioural Trust Management (BTM). Here, the decision whether to trust a service provider depends on the previous behaviour of that provider. When a user interacts with a service provider, he gives feedback on that interaction. Based on this feedback, the BTM engine dynamically computes and updates the reputation of service providers. Users can define behaviour-based trust policies which refer to these reputation values.

In [D5.2], we have defined a trust policy language for BTM. Since trust is a very subjective issue, each user has an individual policy on how to derive trust from feedback. Our trust policy language is flexible enough to support such subjective trust policies. It is based on centrality measures to aggregate feedback into reputation values. The reputation values in turn are used to identify trustworthy service providers.

However, the flexibility of our trust policy language can make it difficult for the user to write policies. We cannot assume that the users are familiar with the formulation of trust policies or that they learn our trust policy language. Requirement 5.9 [D1.2] states that the trust management system shall provide trust policy formulation support. To satisfy this requirement, we assist the users with the formulation of trust policies and help them to understand the effects of their policies. For this purpose, we have developed two tools:

1. **Policy Wizard**:
The policy wizard assists users in the creation of trust policies. It provides a couple of predefined policy blocks, which the users can combine to more complex trust policies.

2. **Trust Visualization**:
Even if a policy wizard is in place, the users have to configure their trust policies by setting parameter values. Our tool helps the users to choose appropriate settings by visualizing the effects of their policies.

9.1 Policy Wizard

The automated processing of trust policies requires that users specify their policies in a formal language. However, ordinary users are not experienced in formal languages. In this section, we describe a tool that assists users in the creation of trust policies. Regarding the design and implementation of such a tool, the following questions arise:

- What kinds of trust policies do users formulate?
- How do policies for service discovery differ from policies for trust decisions?

To answer these questions, we have designed and carried out a user study [44] where participants have formulated trust policies. There were two kinds of policies, one for service discovery and one for trust decisions:

- **Placing policies** specify which service providers the user selects during service discovery.
• **Accepting policies** specify which service providers the user trusts with his personal data.

In total, 23 subjects have participated in the study. The experiment has started after each subject had created a placing and an accepting policy. The subjects could modify their policies during the experiment.

Since we did not want to restrict the space of possible policies, the participants could formulate their policies in natural language. To evaluate the policies, we had to translate them into a formal representation. This allowed us to analyse the expressiveness of our trust policy language. However, the manual translation of trust policies into a formal representation is costly in terms of labour and error-prone because of the ambiguity of natural language. Thus, such a translation is infeasible for enterprise applications. Therefore, we will classify the policies formulated in the study and identify their parameters. We then use the resulting categories to create a ‘policy wizard’ which guides users through the formulation of trust policies.

### 9.1.1 Classification of Trust Policies

We have examined the structure of the policies formulated during the study. Typically, policies were a combination of one or more constraints. A *constraint* is the smallest self-contained statement that specifies a set of service providers. Each constraint has zero or more parameters that specify how the result set is computed. The results of the study show that, on average, each policy consists of 1.26 constraints. We were able to classify the constraints of both placing and accepting policies into six categories:

1. **Trust All** (*All*): Trust all service providers, i.e., send requests to or process requests from all service providers.
2. **Trust None** (*None*): Trust no service provider, i.e., do not send or process any requests.
3. **Probability** (*Prob*): Send requests to or process requests from arbitrary service providers with probability \( r \).
4. **Reputation** (*Rep*): Trust service providers if their reputation (i.e., the percentage of requests processed) exceeds a threshold \( c \).
5. **Number of Requests** (*Req*): Trust service providers who have processed at least \( n \) requests.
6. **Score** (*Scr*): Trust service providers whose score exceeds a threshold \( s \).

Note that only **Reputation** and **Number of Requests** use information on the behaviour of subjects. Since subjects used information on the score very rarely, we omit this category in the following analysis.

Table 1 shows the frequencies of the constraints. First, both **Trust All** and **Trust None** are relatively infrequent as both accepting and placing policies. While **Trust All** is more frequent in placing policies, **Trust None** is more frequent in accepting policies. This is expected because sending requests is cheaper than processing them. Second, there is a considerable difference in the frequency of **Probability** between both types of policies. While it was the most frequent constraint in accepting policies, it was relatively infrequent in placing policies. Third, **Reputation** is the most frequent constraint in placing policies. Note that
Reputation is also quite frequent in accepting policies, but less frequent than Probability. Apparently, subjects found it more important whom to ask than whose requests to process. Finally, Number of Requests is relatively frequent in both kinds of policies.

Table 1: Frequencies of the constraints.

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Accepting Policies</th>
<th>Placing Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trust All</td>
<td>1.6%</td>
<td>7.1%</td>
</tr>
<tr>
<td>Trust None</td>
<td>6.3%</td>
<td>0.8%</td>
</tr>
<tr>
<td>Probability</td>
<td>34.9%</td>
<td>8.7%</td>
</tr>
<tr>
<td>Reputation</td>
<td>27.8%</td>
<td>54.3%</td>
</tr>
<tr>
<td>Number of Requests</td>
<td>29.4%</td>
<td>29.1%</td>
</tr>
</tbody>
</table>

9.1.2 Designing a Policy Wizard

The results of the user study give clear recommendations on the design of a tool that guides users through the creation of trust policies. Such a policy wizard has to fulfil the following design requirements:

- **Support several constraints:**
  The policies created during the study typically consisted of two or more constraints (Probability, Reputation, etc.). These constraints were combined by logic operators, i.e., conjunction and disjunction.

- **Parameterized constraints:**
  Each constraint has a distinct set of parameters (e.g., threshold values) that specify how the constraint is evaluated.

- **Negation of constraints:**
  The tool must allow to negate constraints, i.e. support the NOT operation.

Following these requirements, the creation of a trust policy works as follows:

1. The user selects a constraint from a list of available constraints.
2. He specifies the parameter values of the constraint.
3. To add another constraint, the user chooses a logic operator to concatenate the constraints.
4. The user can negate a constraint by selecting the NOT operation.

In principle, such a tool limits the space of possible policies. However, a tool that fulfils the design requirements is sufficient to generate all policies that participants have formulated during the study.

9.1.3 Available Constraints

We have implemented a policy wizard which fulfils the design requirements specified in the previous section. First, we describe the constraints and parameters supported by our policy wizard. According to Table 1, the wizard has
to support five constraints. Each constraint in turn has at most one parameter. We have consolidated the constraints from the user study and standardized the parameters of the constraints.

The policy wizard supports the following constraints:

- **Accept all:**
  Trust all service providers, i.e., discover all services and always grant access.
  - **No parameters**

- **Reject all:**
  Trust no service provider, i.e., discover no services and never grant access.
  - **No parameters**

- **Probability:**
  Trust a random service provider with a certain probability. For example, grant access in only 50% of the cases.
  - **Threshold:** Threshold value for probability (in per cent).

- **Tit-for-tat:**
  Trust a service provider if he was cooperative in the past, i.e., he received positive feedback.
  - **Threshold:** Specifies the number of negative feedbacks allowed for the service provider.
  - **TimeFrame:** Specifies the number of previous interactions considered for evaluating the condition.
  - **PersonalVsGlobal:** Specifies whether global knowledge or personal knowledge about previous interactions is used.
  - **Optimistic:** Trust a service provider if there is no knowledge about his previous behaviour.

- **Cut-off relative:**
  This condition specifies a cut-off policy considering the reputation of service providers. For example, grant access only to service providers who received at least 50% positive feedback in the past 5 rounds.
  - **TimeFrame:** Specifies the number of previous interactions considered for evaluating the condition.
  - **Type:** Specifies the type of feedback (positive or negative) considered, i.e., the ratio of positive to negative or negative to positive feedback.
  - **PersonalVsGlobal:** Specifies whether global knowledge or personal knowledge about previous interactions is used.
  - **Compare operator:** Specifies the operator (<= or >=) used for the comparison.
  - **Threshold:** Specifies the threshold value used in the comparison, e.g., the percentage of positive feedback.

- **Cut-off absolute:**
  This condition specifies a cut-off strategy with absolute values. It considers the absolute number of positive and negative feedback in previous rounds. For example, grant access only to service providers who received at least 4 positive feedbacks in the past 5 rounds.
- **TimeFrame**: Specifies the number of previous interactions considered for evaluating the condition.
- **Type**: Specifies the type of feedback (positive or negative) considered, i.e., number of positive or negative feedback.
- **PersonalVsGlobal**: Specifies whether global knowledge or personal knowledge about previous interactions is used.
- **Compare operator**: Specifies the operator (\(<=\) or \(>=\)) used for the comparison.
- **Threshold**: Specifies the threshold value used in the comparison, i.e., the number of positive or negative feedbacks.

Each constraint can be represented as a statement of the policy language specified in D5.2. The parameters are represented as placeholders, which are replaced by the values specified by the users. The following example shows the formal representation of the constraint *Probability*, which has the parameter *Threshold*:

```
SELECT id FROM Entity
WHERE random() <= ###Threshold###/100
```

Policies consisting of several constraints must be concatenated by the logic operators AND and OR. The representation of these operators in the policy language is straightforward: AND corresponds to the statement `INTERSECT` and OR corresponds to `UNION`.

![User interface of the policy wizard.](image)

**Figure 13: User interface of the policy wizard.**

The implementation focused on the usability of the tool. Figure 13 shows the user interface of the policy wizard. The upper picture shows an example policy for service discovery, the lower picture for trust decisions.
To add a constraint, the user has to choose from a list of available constraints. The constraints are represented as sentences in natural language. For example, “Select service providers that have received at least 50% positive feedback in the past 5 interactions”. The underlined words represent parameters which the user may change.

The wizard shows the number of providers that respect the current policy to give the user an idea of how strict his policy is. If no providers are considered trustworthy, the user is asked to change the parameters of his policy.

### 9.2 Trust Visualization

The trust policy language defined in D5.2 is based on centrality measures. Centrality measures are a well-established approach for analysing network structures such as feedback graphs. They are used to compute the importance of a vertex based on the graph structure. In our setting, users and service providers are represented as vertices and the feedback as edges. Centrality measures can then be used to compute the reputation of service providers. They compute a numerical value, the centrality score, for each service provider which allows for a ranking of the providers. The intuition is that service providers with high centrality scores are considered trustworthy. Users can refer to the centrality scores in their trust policies.

In this section, we consider the following application scenario:

*Alice is seeking a job placement. She registers with a placement coordinator and selects a suitable programme. To register with the programme, Alice has to submit personal data such as her CV. Alice specifies how her data is used by formulating the following trust policy: “I trust service providers whose centrality score is among the top five of all providers.” Suitable service providers are retrieved and ranked according to their trust score. If no service providers were discovered, Alice has to relax her trust policy in order to increase the number of results. Alice is presented a list of suitable service providers. She selects a service provider and approves that her personal data is released to the provider.*

Research has proposed a number of centrality measures, see [50] for an overview. Centrality measures can be categorized as according to their algorithmic complexity [40]:

- **Local measures** take into account only direct neighbors of a node. Examples are Indegree (the sum of the weights of incoming edges) and Outdegree (the sum of the weights of outgoing edges). Local measures have linear computational complexity.

- **Eigenvector-based measures** not only consider the incoming edges, but also the centrality values of the neighbors. In other words, nodes may 'pass' their score to their neighbors. Examples are PageRank [47] and HITS (Hubs and Authorities) [45]. Eigenvector-based measures have quadratic complexity.

- **Distance-based measures** compute the shortest paths between all pairs of nodes. Examples are Closeness [49] and Betweenness [41]. Distance-based measures have cubic complexity.
Each centrality measure is designed for a specific context and satisfies specific information needs. Users have to experiment with different centrality measures to find a suitable measure for their trust policies. Our tool helps users to do so by visualizing the centrality scores. This allows users to assess the effects of their policies.

We have developed an operational visualization tool called SONAR [43], which is freely available on http://github.com/dsp/projectsonar/ under the GPL license. The results of a performance evaluation show that our tool scales well with the number of users and adds only minimal overhead to the centrality measures computation.

9.2.1 Implementation

In this section, we present the requirements, features and architecture of our visualization tool. Regarding the analysis and visualization of trust policies, the following problems arise:

P1 Research has proposed dozens of centrality measures, and we expect more in the future. Thus, the architecture of our tool has to be extensible to allow for the addition of novel centrality measures.

P2 End users have different analysis needs than security architects. Architects typically examine the feedback graph as a whole. In contrast, users are interested in the subgraph of a particular service provider.

Answering to these problems is difficult because of the following technical challenges:

C1 Feedback graphs are typically very large and the computation of centrality measures is expensive. Thus, our visualization tool must be scalable.

C2 Most existing tools for network analysis are standalone desktop applications [42]. However, the service-oriented architecture of TAS³ requires our visualization tool to be available as a web application.

9.2.1.1 Requirements

First, we present the requirements for our visualization tool. Considering the application scenario of behavioural trust policies, there are functional and technical requirements that our visualization tool must fulfil. From problems P₁ and P₂, the following functional requirements arise:

Rₑ Extensibility: Centrality measures are improved continually, and new measures keep coming up. Thus, the application must allow both the modification of the centrality measures implemented and the addition of novel centrality measures.

Rᵤ User Centricity: Users might only be interested in a particular subgraph. Further, users want to experiment with different centrality measures in their analysis. Hence, the application must support user centricity by letting users make customized analyses based on a given subgraph.

Challenges C₁ to C₂ lead to the following technical requirements:
RS  *Scalability*: Users might not be able to perform the analysis on their machines due to lack of resources. Thus, computation and visualization should be separated.

RI  *Integrability*: The visualization tool must be available as a web application in order to integrate into existing TAS³ components.

We satisfy these requirements by developing a Web-based tool for the analysis and visualization of trust policies. While our tool already supports a set of commonly used centrality measures, a plug-in API allows developers to add new centrality measures on the fly. Our visualization tool features a user-centric view that allows each user to analyse the subgraph of a particular service provider. To cope with expensive computations on large feedback graphs, our visualization tool computes the centrality measures on dedicated servers. A Web browser which runs on the client visualizes the results of the analysis. The Web-based architecture allows various TAS³ components to embed our visualization tool directly.

Figure 14 shows the user interface of our visualization tool. The feedback graph shown results from the following use case:

*Alice wants to analyse her trust policy which consist of the measures Indegree and Betweenness. She specifies that Indegree is visualized as size of the nodes and Betweenness as distance to the centre of the graph. The analysis should consider time frame $t = [52, 56]$ and 2 hops of the feedback graph.*

![Figure 14: User interface of the tool for trust visualization.](image)

### 9.2.1.2 Features

We now describe the features of our visualization tool and explain how it meets the requirements.

**Plug-in API**: In order to fulfil RS, we have developed a plug-in API which simplifies the addition of new centrality measures. Minimal knowledge of the inner workings of our visualization tool is needed to write new centrality measures. A developer only has to implement the actual algorithm. By inheriting...
our tool’s plug-in class, the centrality measures have access to the feedback graph. The results of the centrality measures are returned as annotations of the edges or nodes. As soon as the new centrality measures class is accessible to the Java classloader, our visualization tool deploys it on the fly. Our visualization tool runs on a Java Servlet container, so centrality measures can be written in Java as well as in Java-based functional languages like Scala or Clojure. We have included a set of commonly used centrality measures plug-ins into our visualization tool: In-and Outdegree, Betweenness, Closeness and PageRank.

User-Centric View: Our visualization tool features two different views:

(1) Security architects are able to calculate centrality measures based on the global graph, i.e., the entire feedback graph consisting of all providers and all feedbacks. They can combine different centrality measures by specifying how the individual measures are visualized as colour, size or distance of the nodes and edges. To handle large graphs, architects can also specify the time frame of their analysis as well as a number of nodes to be displayed. Then, the centrality measures calculation takes into account nodes and edges that existed within the time frame, and displays only the most central nodes.

(2) End users are able to limit the analysis to the local subgraph of a given service provider. Users define this subgraph by specifying a provider and the number of hops they want to analyse. Except for the limited graph, users have the same analysis features as architects. Our visualization tool also supports the exploration of the feedback graph by displaying information about each node and edge (see Figure 14). In the example, user foobar has a Betweenness of 18.5 and an Indegree of 3.0. The user-centric view fulfils RU.

Caching and Load Balancing: To handle large graphs and to serve many requests, our visualization tool has been developed with scalability aspects in mind. Our visualization tool features the caching of the feedback graph as well as of the centrality calculation. Our visualization tool executes costly calculations on dedicated servers, while all visualizations are done by the clients using a JavaScript-based rendering approach. Since centrality calculation in our visualization tool is stateless, it is possible to set up a load balancer which distributes the requests among the available servers. The caching and load balancing satisfy Rs.

Flexible Data Model: Our visualization tool features a directed multi-graph with annotatable nodes and edges as a universal representation of the feedback graph. We decided to separate the annotations of nodes and edges from the actual structure of the graph. This allows each node and edge to have several annotations (i.e., centrality values), which the individual plug-ins compute. The feedback graph is stored in a relational database management system. An important design decision of ours is to use an object-relational mapping (ORM) between the relational graph database and our visualization tool’s multigraph. We chose Hibernate as ORM library because it enables configuring the database mapping via XML files.

Customizability: A Java-Script GUI provides easy customizability and thus allows embedding our visualization tool into existing TAS³ components. Further, our visualization tool is open source software: If necessary, developers can adjust
every component to conform to an existing user interface. We conclude that our visualization tool meets RI.

9.2.1.3 Architecture
An important design decision has been to implement our visualization tool as a Web application using the Google Web Toolkit (GWT). We chose GWT because it provides a powerful infrastructure for high performance Web applications. As illustrated in Figure 15, the architecture of our visualization tool is split into three parts resembling the Model-View-Presenter (MVP) [48] design pattern. We favour MVP over the classic Model-View-Controller because it features a strong separation between the data model and calculation logic on one hand and the user interface on the other.

![Figure 15: Software architecture of the visualization tool.](image)

**Model** (Server): The model contains the data to be displayed in the user interface. The server keeps track of all queries, caches incoming requests and responses, communicates with the database, loads, and runs the centrality measures plug-ins. The server processes requests as follows: (1) The Remote Procedure Call (RPC) system reacts to client requests by dispatching the requests to the centrality calculation unit. (2) The server fetches the specified subgraph $S$ using the database interface. In our use case, the server traverses the graph starting with Alice. All nodes $N$ that are at most 2 hops away from Alice and that existed during time frame $t$ are part of subgraph $S$. All edges between nodes $N$ from the original graph that existed during time frame $t$ are part of $S$ as well. (3) The centrality measures Plug-in System loads the specified plug-ins, i.e., Betweenness and Indegree, in our use case. (4) The server provides access to subgraph $S$ to all plug-ins, which calculate the centrality measures. (5) The RPC system returns the annotated subgraph to the client.

**Presenter** (Client): The presenter retrieves data from the model and prepares it for the view component. It runs as JavaScript executable within the user’s Web browser. The RPC Handler submits asynchronous RPC requests to the server. It receives the annotated subgraph from the server and forwards it to the Graph Converter. Events are dispatched through a central Event Bus, implementing the author-subscriber design pattern. We decided to use an event bus to avoid
dependencies between the modules and to decouple the layers. This separation of
duties enables easy unit testing of the presenter and the view. To abstract from
asynchronous server requests, all inter-module communication is done via the
event bus. The GraphConverter transforms the subgraph into a visualizable
structure. This data structure contains specific information needed to display the
feedback graph, such as the position of each node, its size and its colour. As this
information is bound to the specific implementation of the client, it is kept away
from the server. In our use case, the Graph Converter maps the Indegree of a
node to its size and the Betweenness of a node to its distance to the centre of the
graph. The GUI Controller module coordinates the functionality of the client.

**View** (Client): The view displays data from the presenter and forwards user
events to it. It is responsible for displaying GUI elements and for drawing the
subgraph within the user's Web browser. We decided to encapsulate the
GraphDrawer module to make it interchangeable, enabling different kinds of
graph renderings. As graph renderer, we chose JSXGraph, which provides a
powerful API combined with an excellent rendering performance.

### 9.2.2 Performance Evaluation

To be useable in a productive environment, our visualization tool has to serve
requests to a large number of users at the same time. It is critical to ensure short
response times of the application as well as a high level of scalability. We have
used common performance testing techniques to evaluate the runtime behaviour
and scalability of our visualization tool under heavy load.

As server we have used a Sun Microsystem M 24 Workstation with an Intel Quad
Core 3 GHz processor, 12 MB L2 cache and 4 GB RAM. To generate the load, we
have used two ordinary computers connected to the server via 1 GBit switched
Ethernet. We have simulated 20 to 200 concurrent users and their click path
through the application using Apache JMeter. We have added the users one by
one before we recorded the results so that both our application and the server
were able to fill up their caches. We have repeated each test run 30 times to
exclude random effects.

The performance test simulated a user who gets (1) the list of available centrality
measures as well as (2) the list of users, and calculates two different graphs, (3) a
global graph with the 30 most central nodes according to the Indegree measure
as well as (4) a user-centric graph representing the friends and friends of a friend
(2 hops) of a random user according to the PageRank measure.

Figure 16 shows the average run times of the test cases. The run times increase
linearly from 5 ms to 60 ms. We notice that our visualization tool scales well with
the number of users. The server reached its capacity at 200 concurrent users. The
cost for getting the available centrality measures and users (i.e., the overhead of
our application) is dominated by the cost for calculating the centrality measures
on the graphs. Thus, we conclude that our application adds only minimal
overhead to the computation of the centrality measures.
Figure 16: Results of the evaluation of the visualization tool.
9.2.3 Tutorial

Finally, we include a tutorial on how end users can use our visualization tool to test their trust policies. Figure 17 shows a global graph consisting of 194 nodes. The user has selected three centrality measures:

1. The PageRank of a node is visualized as its size.
2. Edge Betweenness is visualized as the width of an edge.
3. The colour of a node represents its Node Betweenness.

Note that more centrality measures can be defined and/or selected.

Our visualization tool features restrictions on the visualized graph, i.e., the history of the data as well as the number of nodes to be displayed. With the scrollbar at the bottom of the screen, users can define during which time frame, i.e., period of time, nodes and edges had to exist so that the analysis takes them into account. In Figure 18, only nodes and edges that existed between round 51 and 54 have been taken into account. Another option is limiting the number of nodes in the graph. Only the ten most central nodes are being displayed. Further, the user has selected the following centrality measures:

1. Node Betweenness is visualized as the size of a node.
2. The PageRank of a node is visualized as its colour.
3. The width of an edge represents its Edge Betweenness.
Figure 19 shows a user-centric graph. The node of user ‘Arnoe’ is in the centre of the graph. Its contacts and contacts of contacts (with maximal 2 hops distance) are arranged around the node of the user. The user has selected the following centrality measures:

1) The PageRank of a node is visualized as its distance to the centre of the graph.
2) Edge Betweenness is visualized as the width of an edge.
3) The colour of a node represents its Node Betweenness.
4) The Outdegree of a node is visualized as its size.

Our visualization tool also supports the exploration of the feedback graph. If the user moves his mouse over an edge or a node, our visualization tool displays information about the item: In Figure 19, our visualization tool displays information about node ‘Kopflocke’: It has a PageRank of 0.83, a Node Betweenness of 0.23, and an Outdegree of 23 in the current graph. The graphs our visualization tool generates are not static. All nodes can be moved by drag and drop (as long as no centrality measure is mapped to distance to the centre).
Figure 19: User-centric graph.
10 Conclusion

In this research report we presented various novel trust metrics that try to overcome the limitation of the existing solutions in terms of flexibility, adaptability and innovation. We proposed to use KPIs parameters to build a flexible and federated trust model in which every user will be able to adapt the trust metric to his proper trust objectives. This solution breaks with the incertitude due to the subjectivity of traditional trust models, by relying on quantifiable performance values that can be adapted to the personal trust perception of every user. We also proposed a new trust metric that takes advantage of the audit logs for security testing systems. We demonstrated how audit data can be analyzed in order to evaluate the trust level of a tested entity. This trust metric reflects the resilience of a system against security failures. We also present a new solution for the evaluation of the credential based trust model that uses an ontological representation of the credential hierarchy and interaction. Next, we addressed one crucial problematic that affects the new internet applications and particularly the Web 2.0 applications in which tremendous quantities of personal and private information are exposed without any control or trust evaluation. We end up with a novel trust metric based on negative feedbacks that evaluates the trustworthiness of an entity dealing with private and personal information. This system is able to track personal data and sanction the entity that is not compliant with his privacy policy. Finally, we have presented two tools that increase the usability of trust policies: (1) Based on a user study, we have developed a policy wizard that assists users in the creation of trust policies. (2) Our trust visualization tool helps users to find suitable reputation metrics for their trust policies and to assess the effects of their policies.
11 References


[D1.1] TAS³ Deliverable D1.1. Requirements Analysis: State of the Art and Technology Assessment

[D1.2] TAS³ Deliverable D1.2. Requirements Assessment Report

[D2.1] TAS³ Deliverable D2.1. TAS³ Architecture

[D5.1] TAS³ Deliverable D5.1. Trust Management Architecture Design

[D5.2] TAS³ Deliverable D5.2. Behavioral Trust Management Engine

[D9.1] TAS³ Deliverable D9.1. Pilots Specifications and Use Case Scenarios

[D10.2] TAS³ Deliverable D10.2. Trustworthiness Architecture and Proof of Concept
## Amendment History

<table>
<thead>
<tr>
<th>Ver</th>
<th>Date</th>
<th>Author</th>
<th>Description/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2009-10-07</td>
<td>JH</td>
<td>First design from scratch.</td>
</tr>
<tr>
<td>0.2</td>
<td>2009-10-08</td>
<td>JH</td>
<td>Added various extra sections such as Refs.</td>
</tr>
<tr>
<td>0.3</td>
<td>2009-10-23</td>
<td>JH</td>
<td>Brought more in line with LaTeX. CodeBlock style added, some touchups.</td>
</tr>
<tr>
<td>1.0</td>
<td>2010-03-31</td>
<td>ST</td>
<td>Ready for submission</td>
</tr>
<tr>
<td>1.1</td>
<td>2011-09-20</td>
<td>CH</td>
<td>Added section on trust usability</td>
</tr>
<tr>
<td>1.2</td>
<td>2011-09-30</td>
<td>CH</td>
<td>Revised trust usability</td>
</tr>
<tr>
<td>1.3</td>
<td>2011-10-12</td>
<td>CH</td>
<td>Implemented comments by De Angelis</td>
</tr>
<tr>
<td>2.0</td>
<td>2011-10-17</td>
<td></td>
<td>Final version second iteration</td>
</tr>
</tbody>
</table>